IMAGETEAM™ 4X00 Series

For Adaptus Imaging Technology Imagers:
IT4000, IT4100, and IT4300

Software Development Kit (SDK)

User's Guide

Disclaimer

Hand Held Products, Inc. d/b/a Hand Held Products (“Hand Held Products”) reserves the right to make changes in specifications and other information contained in this document without prior notice, and the reader should in all cases consult Hand Held Products to determine whether any such changes have been made. The information in this publication does not represent a commitment on the part of Hand Held Products.

Hand Held Products shall not be liable for technical or editorial errors or omissions contained herein; nor for incidental or consequential damages resulting from the furnishing, performance, or use of this material.

Web Address: www.handheld.com

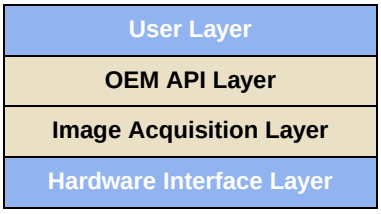
Table of Contents

1 Introduction

This document is an overview of the structure of the IT4X00 Series OEM imaging software provided by Hand Held Products. Detailed information that describes the Hand Held Products OEM Application Programming Interface is provided as a part of the 4X00 Series.

Design Overview

The Hand Held Products OEM imaging software supports a number of different Hand Held Products imaging engines, while exposing the user to a common programming interface for all supported imagers. This approach provides Hand Held Products with a simple way of adding support for new imaging hardware, while allowing user software to remain largely unchanged. In support of this design approach, the software components implemented in the Hand Held Products OEM imaging software have been organized in the following layered format:



Of the four layers, Hand Held Products supplies the OEM API and Image Acquisition layers.

User Layer

The User Layer can be the application that is written by a developer or end user that accesses the imaging system by using the OEM API functions. The User Layer can alternatively be an abstraction layer provided by an OEM that allows the OEM to customize the level of API functionality exposed to their end user. Either option has access to all the applicable 4X00 Series functions that access and control the imaging subsystem

OEM API Layer

The OEM API Software is supplied by Hand Held Products and is the primary interface to the imaging system. For Windows CE© applications, the OEM API Software functionality is provided in the form of a Windows Dynamic Link Library. For non-Windows CE applications, the form of the OEM API is to be determined. Full explanation of the 4X00 Series is provided later in this document.

Image Acquisition Layer

The Image Acquisition Software layer is the software supplied by Hand Held Products that is responsible for imager auto detection, initialization, state management, exposure control, and image acquisition. During application execution, this software maintains the state and control of the imager, processes requests for images, and executes configuration and control requests. For Windows CE applications, this layer is implemented in a stream device driver and is used by the OEM API layer. For non-Windows CE applications, the form of this layer and its interface to the OEM API layer is to be determined.

Hardware Interface Layer

The Hardware Interface Software is the hardware-specific software provided by the system developer and used by the Image Acquisition Software to access the physical imaging hardware and run hardware-specific tasks on the system. This software is called upon by Image Acquisition Software to handle physical control of the hardware, such as communication with the imager, timing functionality, low level interrupt service routine control, DMA initialization and control, system specific memory control, and illumination functionality. For Windows CE applications, this layer is provided in the form of a Windows Dynamic Link Library that is loaded by the Imaging device driver. For non-Windows CE applications, the form of this layer and its interface to the Image Acquisition Software is to be determined.

Features of the 4X00 Series

The 4X00 Series consists of the following:

- The API Definition and Documentation
- API Libraries
- Sample Code

The 4X00 Series functions are defined on a higher level so they can be easily understood and integrated into your applications.

- The image/data capture engine is easily integrated.
- A single API is used for all Hand Held Products engines.
- Libraries are available for Windows CE 2.x and Windows CE 3.x.
- Sample source code is provided so you can see an example of how to use the API functions.

Target Operating Systems for the 4X00 Series

The 4X00 Series is designed for use with Microsoft Windows CE version 2.0 and above

Image Engine API Library Summary

The following is a summary of the API functions. The full description of each of function is found on the page noted.

Core Function	*Summary Description*	*Page*
oemAimerOn	Turn on/off the engine's aiming LEDs	2-1
oemConnect	Initialize connection with engine device	2-2
oemDisconnect	Close the connection to the engine device	2-3
oemGetAPIRevision	Retrieve the API library's software revision information	2-4
oemGetDecodeMode	Retrieve the decoding mode of the engine	2-5
oemGetErrorMessage	Given an error number, returns a string describing the error condition	2-6
oemGetImagerProperties	Returns information about the imager	2-9
oemGetLeaveLightsOn	Return parameter reflecting operation mode of illumination LEDs during scanning	2-11
oemGetScanDriverRevision	Returns the revision of the scan driver component that interfaces to the OEM API layer	2-13
oemLeaveLightsOn	Configure illumination LEDs to be always on or in normal mode during scanning	2-43
oemPowerOffImager	Allows the application to fully power down the imager for additional power control	2-43
Decoder Functions		
oemDefaultSymbology	Return some or all symbology options back to factory default settings	2-2
oemDisableSymbology	Disable decoding of specific bar code symbologies	2-2
oemDisableSymbologyAll	Disable decoding of all symbologies, including unknown symbologies	2-3
oemEnableSymbology	Enable decoding of specific bar code symbologies	2-3
oemEnableSymbologyAll	Enable decoding of all symbologies, including unknown symbologies	2-3
oemGetDecodeAttemptLimit	Retrieves the current decode attempt maximum time limit	2-4
oemGetDecodeCenteringWindow	Enable decode centering mode	2-4
oemGetDecodeTime	Returns time to decode in milliseconds	2-5
oemGetDecoderRevision	Retrieve decoder's current revision	2-5
oemGetLinearRange	Get the size of the window used in Advanced Linear decoding mode	2-12
oemGetMaxMessageChars	Retrieve the size of the largest possible message (in characters)	2-12
oemGetPrintWeight	Retrieve the current or default print weight the decoder expects	2-12
oemGetSearchTimeLimit	Retrieves the current search maximum time limit	2-13
oemGetSetupAll	Gets all symbology decoding options	2-13
oemGetSetupAusPost	Gets the Australian Postal Code symbology decoding options	2-14
oemGetSetupAztec	Gets the Aztec and Aztec Mesa Code symbology-specific decoding options	2-14
oemGetSetupBPO	Gets the British Postal Code symbology decoding options	2-14
oemGetSetupCanPost	Gets the Canadian Postal Code symbology decoding options	2-15
oemGetSetupChinaPost	Gets the Chinese Postal Code symbology decoding options	2-15

oemGetSetupCodabar	Gets the Codabar symbology decoding options	2-16
oemGetSetupCodablock	Gets the Codablock symbology decoding options	2-16
oemGetSetupCode11	Gets the Code 11 symbology decoding options	2-17
oemGetSetupCode128	Gets the Code 128 symbology decoding options	2-18
oemGetSetupCode16K	Gets the Code 16K symbology decoding options	2-18
oemGetSetupCode32	Gets the Code 32 symbology decoding options	2-19
oemGetSetupCode39	Gets the Code 39 symbology decoding options	2-19
oemGetSetupCode49	Gets the Code 49 symbology decoding options	2-20
oemGetSetupCode93	Gets the Code 93 symbology decoding options	2-21
oemGetSetupComposite	Gets the EANoUCC Composite symbology decoding options	2-21
oemGetSetupCompositeEx	Gets the EANoUCC Composite as well as other Composite symbology decoding options	2-22
oemGetSetupCouponCode	Gets the Coupon Code symbology decoding options	2-23
oemGetSetupDataMatrix	Gets the Data Matrix symbology decoding options	2-23
oemGetSetupDutchPost	Gets the Dutch Postal Code symbology decoding options	2-24
oemGetSetupEAN8	Gets the EAN 8 symbology decoding options	2-24
oemGetSetupEAN13	Gets the EAN 13 symbology decoding options	2-25
oemGetSetupIATA25	Gets the IATA 2 of 5 symbology decoding options	2-26
oemGetSetupInt25	Gets the Interleaved 2 of 5 symbology decoding options	2-27
oemGetSetupISBT	Gets the ISBT symbology decoding options	2-28
oemGetSetupJapost	Gets the Japanese Postal Code symbology decoding options	2-28
oemGetSetupKoreanPost	Gets the Korean Postal Code symbology decoding options	2-28
oemGetSetupMaxicode	Gets the MaxiCode symbology decoding options	2-29
oemGetSetupMesa	Gets the Aztec Mesa symbology decoding options	2-30
oemGetSetupMicroPDF	Gets the MicroPDF417 symbology decoding options	2-30
oemGetSetupMSI	Gets the MSI symbology decoding options	2-31
oemGetSetupMx25	Gets the Matrix 2 of 5 symbology decoding options	2-32
oemGetSetupOCR	Gets the OCR symbology decoding options	2-32
oemGetSetupPDF417	Gets the PDF417 symbology decoding options	2-33
oemGetSetupPlanet	Gets the Planet Code symbology decoding options	2-34
oemGetSetupPlessey	Gets the Plessey symbology decoding options	2-34
oemGetSetupPosiCode	Gets the Posicode symbology decoding options	2-35
oemGetSetupPostnet	Gets the Postnet symbology decoding options	2-36
oemGetSetupQR	Gets the QR Code symbology decoding options	2-36
oemGetSetupRSS	Gets the RSS symbology decoding options	2-37
oemGetSetupStrt25	Gets the Straight 2 of 5 symbology decoding options	2-37
oemGetSetupTelepen	Gets the Telepen symbology decoding options	2-38
oemGetSetupTLC39	Gets the TLC39 symbology decoding options	2-38
oemGetSetupTrioptic	Gets the Trioptic symbology decoding options	2-39
oemGetSetupUPCA	Gets the UPC version A symbology decoding options	2-39
oemGetSetupUPCE	Gets the UPC version E0 & E1 symbology decoding options	2-40
oemGetVideoReverse	Determines if decoding of inverted symbols is enabled	2-41
oemLightsOn	Turns the engine's illumination LEDs on and off.	2-43
oemSetDecodeAttemptLimit	Sets the decode attempt maximum time limit.	2-44
oemSetDecodeCenteringWindow	Enables/Setup decode centering mode.	2-44

oemWaitForDecodeRaw	The engine scans until a symbol is decoded, or a timeout is reached - decoded message is returned in raw form	2-68
oemWaitMultipleDecode	Reads multiple symbols using a single function call	2-69
oemWaitMultipleDecodeRaw	Reads multiple symbols using a single function call - decoded message is returned in raw form	2-70
Imaging Functions		
oemAcquireImage	Acquires an image but does not return it.	2-1
oemGetExposureSettings	Retrieves image parameters used during image acquisition.	2-6
oemGetImage	Retrieve an image from the engine	2-7
oemGetImageData	Get image data from the Imager	2-8
oemGetImagerInfo	Retrieve the pixel dimensions and bit depth of the engine's Imager	2-8
oemGetLastImage	Retrieves the last image acquired	2-10
oemGetLastImageExt	Retrieves the last image and the exposure parameters acquired by the image engine	2-10
oemGetLastImageSize	Retrieve the rows, columns and size of the last image returned by oemAcquireImage, oemGetImage, or oemImageStreamRead.	2-11
oemGetSetupImager	Gets the current imager setup values.	2-27
oemImageStreamInit	Intialize image streaming interface.	2-42
oemImageStreamRead	Retrieve current image	2-42
oemImageStreamStart	Start imager acquiring images	2-42
oemImageStreamStop	Tell imager to stop acquiring images	2-43
oemStartIntellImgXfer	Starts an IQ image transfer	2-65

Data Type Definitions

Throughout this document the following variable types are used. These data types are defined in the Oemdecodece.h header file.

Variable	*Description*
BOOL	OS-dependent-size Boolean variable (1 = true, 0 = false).
BYTE	8 bit unsigned variable.
DecodeMsg_t	Typedef structure used to define decoded bar code message information.
DecodeMsgRaw_t	Typedef structure used to define decoded bar code message information in raw format.
DWORD	32 bit unsigned integer variable.
ExposureMode_t	Enumerated integer type used to select the exposure mode used during image acquisition. The mode options are: fixed, on chip, and Hand Held Products exposure mode.
ExposureSettings_t	Typedef structure used to hold all possible exposure setting parameters used during image acquisition.
FileFormat_t	Enumerated integer type identifying possible image data formats.
ImagerDesc_t	Typedef structure used to set the format of the images returned from the engine.
ImagerSetup_t	Typedef structure used to set imager parameters used during image acquisition.
IntellImgDesc_t	Typedef structure used to set parameters used during an IQ image transfer.
OCRDirection_t	Enumerated integer identifying OCR character orientation.
OCRMode_t	Enumerated integer identifying the OCR font to be decoded.

Variable	*Description*
Result_t	Enumerated integer type that defines API function result values. See Function Result Values on page 3-2.
ScanIlluminat_t	Enumerated integer type that identifies possible illumination modes used during image acquisition.
SetupType_t	Enumerated integer type that identifies setup type for configuration functions.
TCHAR	OS-dependent character variable. 16 bit for Unicode systems, otherwise 8 bits.
WORD	16 bit unsigned integer variable.

2 API Function Descriptions

The following is an alphabetic listing of each API function with its complete description and a prototype for each function. All API functions return a result code of type Result_t. See Function Result Values on page 3-2 for a table of result code values.

oemAcquireImage

This function tells the Imager to acquire an image, but does not return the image.

Result_t ***oemAcquireImage (***
const ImagerSetup_t *pImagerSetup
)

Return Values

RESULT_SUCCESS
RESULT_ERR_UNSUPPORTED
RESULT_ERR_NOIMAGE
RESULT_ERR_PARAMETER

Parameters

pImagerSetup

Pointer to a data structure that sets up the Imager.

```
typedef struct
{
   int NumUpdates;
   int Exposure;
   int Gain;
   int TargetWhite;
   int TargetWhiteWindow;
   int Reserved;
      } ImagerSetup _t;
```

NumUpdates: The Imager takes multiple images attempting to reach a target white value in the image, while adjusting its gain and exposure settings. This limits the number of attempts.

Exposure : Maximum allowable exposure setting for the Imager. The greater the exposure, the more light, and hence the brighter the image. However, it also increases motion sensitivity.

Gain: Maximum allowable gain for the Imager. Higher gain will yield a brighter image, at the expense of added noise. This will not affect motion sensitivity.

TargetWhite: The Imager takes multiple images attempting to reach this target white value in the image, plus or minus the TargetWhiteWindow value.

TargetWhiteWindow: The acceptable target white value falls within the range TargetWhite +/- TargetWhiteWindow.

Reserved: Must be set to -1.

oemAimerOn

This function turns the engine's aiming mechanism on or off.

Result_t ***oemAimerOn (***
BOOL bEnable
)

Return Values

RESULT_SUCCESS
RESULT_ERR_DRIVER
RESULT_ERR_NORESPONSE

Parameters

bEnable

If TRUE, the aiming mechanism is turned on; otherwise the aiming mechanism is turned off.

oemConnect

The application should call this function before any other API functions. Once an application has connected to the engine, all other API functions can be successfully called. The application does not need to re-connect to the engine unless it has called oemDisconnect (page 9).

Result_t oemConnect ( )

Return Values

RESULT_SUCCESS
RESULT_ERR_DRIVER

Parameters

All Reserved, should be NULL

oemDefaultSymbology

This function sets the specified symbologies to their factory default configurations. See the individual setup functions to determine the factory default setting for a particular symbology.

Result_t oemDefaultSymbology (
BOOL *pSymbology,
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

pSymbology

Points to an array of symbologies you want to default. For example, if a value is set to 1 (i.e., pSymbology [SYM_AZTEC]=1), then Aztec symbology is defaulted.

Note: The array must be of size MAX_SYMBOLOGIES (Defined in Oemdecodece.h). You must initialize the entire array before calling this function.

oemDisableSymbology

This function disables specified symbologies from decoding.

Result_t oemDisableSymbology (
BOOL *pSymbology
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

pSymbology

Points to an array of symbologies you want to disable. For example, if a value is set to 1 (i.e., pSymbology [SYM_AZTEC]=1), then Aztec decoding is disabled.

Note: The array must be of size MAX_SYMBOLOGIES (Defined in Oemdecodece.h). You must initialize the entire array before calling this function.

oemDisableSymbologyAll

This function disables all symbologies from decoding, including any unknown symbologies.

Result_t ***oemDisableSymbologyAll (***
void
)

Return Values

RESULT_SUCCESS

Parameters

None.

oemDisconnect

This function terminates the connection with the engine. Any resources used by the connection device driver are freed.

Result_t ***oemDisconnect (***
void
)

Return Values

RESULT_SUCCESS

Parameters

None.

oemEnableSymbology

This function enables specified symbologies for decoding.

Result_t ***oemEnableSymbology (***
BOOL *pSymbology
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

pSymbology

Points to an array of symbologies you want to enable. For example, if a value is set to 1 (i.e., pSymbology [SYM_AZTEC]=1), then Aztec decoding is enabled.

Note: The array must be of size MAX_SYMBOLOGIES (Defined in Oemdecodece.h). You must initialize the entire array before calling this function.

oemEnableSymbologyAll

This function enables all symbologies for decoding, including any unknown symbologies.

Result_t ***oemEnableSymbologyAll (***
void
)

Return Values

RESULT_SUCCESS

Parameters

None.

oemGetAPIRevision

This function returns an ASCII string containing the API's current revision.

Result_t ***oemGetAPIRevision (***
TCHAR *pszRev
)

Return Values

RESULT_SUCCESS always

Parameters

pszRev

Upon successful return, this null-terminated string is filled in with the revision level of the API. The caller must allocate at least ENGINE_API_RESPONSE_LEN bytes for this string.

oemGetDecodeAttemptLimit

This function is used to retrieve the current decode attempt maximum time limit. The limit, specified in milliseconds, is the maximum amount of time the decoder may use to attempt a decode on the current image.

Result_t ***oemGetDecodeAttemptLimit (***
SetupType_t SetupType,
WORD *nLimit
)

Return Values

RESULT_SUCCESS
RESULT_ERR_UNSUPPORTED
RESULT_ERR_PARAMETER

Parameters

Setup Type

SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

pnLimit

Points to a word variable that upon return of RESULT_SUCCESS will contain the decode attempt maximum time limit. A value of zero indicates no limit.

oemGetDecodeCenteringWindow

This function returns the Enabled/Setup information for decode centering mode. In this mode, a decode call is only successful if the area bounding the decoded symbol intersects a caller defined rectangle located about the center of the captured image.

Note: This function allows the engine to discriminate symbols that are located physically close to each other so only one symbol is captured during decode. Only the symbol intersecting the intersection rectangle is returned.

Result_t ***oemGetDecodeCenteringWindow (***
SetupType_t SetupType,
BOOL *pbEnabled,
RECT *pIntersectRect
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

SetupType

SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

PbEnabled

Pointer to a BOOL which returns TRUE if centering mode enabled or FALSE if it's not.

oemGetDecodeMode

This function retrieves the decoding mode of the engine.

```
Result_t    oemGetDecodeMode (
            SetupType_t SetupType,
            WORD *pnMode
            )
```

Return Values

RESULT_SUCCESS
RESULT_ERR_UNSUPPORTED

Parameters

SetupType

SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

pnMode

Points to a WORD variable that will be filled in with status for the decode mode:
1 = Standard
2 = Advanced Linear
4 = Quick Omni

oemGetDecoderRevision

This function returns an ASCII string containing the decoder's current revision.

```
Result_t    oemGetDecoderRevision (
            TCHAR *pszRev
            )
```

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

pszRev

Upon successful return, this null terminated string is filled with the revision level of the decoder software. The caller must allocate at least ENGINE_API_RESPONSE_LENGTH bytes for this string.

oemGetDecodeTime

This function the time in milliseconds that it took to decode the barcode data returned by oemWaitForDecode().

```
Result_t    oemGetDecodeTime  (
            DWORD *pdwTime,
            )
```

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

pdwTime

Points to an unsinged 32 bit integer into which the last time to decode is returned.

oemGetErrorMessage

This function returns a string containing the description of the error corresponding to the nError parameter.

Result_t ***oemGetErrorMessage (***
TCHAR *pszErrorMsg,
Result_t nError
)

Return Values

RESULT_SUCCESS always

Parameters

pszErrorMsg

Upon successful return, this null-terminated string is filled in with text describing the error identified by the nError parameter. The caller must allocate at least ENGINE_API_RESPONSE_LEN bytes for this string.

nError

A value returned from another API function call.

oemGetExposureSettings

This function is used to retrieve the various image parameters that are used during image acquisition.

Result_t ***oemGetExposureSettings (***
ExposureSettings_t *pExpsoureSettings
)

Return Values

RESULT_SUCCESS
RESULT_ERR_UNSUPPORTED
RESULT_ERR_DRIVER
RESULT_ERR_PARAMETER

Parameters

pExposureSettings

A pointer to an exposure settings structure that has been initialized to zero. Upon return the structure will contain the current exposure parameters used during image acquisition.

oemGetImage

This function retrieves an image from the engine and stores it in memory pointed to by pImageBuffer.

Result_t oemGetImage (
BYTE *pImageBuffer,
DWORD *pdwSize,
WORD nTop,
WORD nLeft,
WORD nRight,
WORD nBottom,
WORD nSkip,
WORD nBits,
FileFormat_t nFormat,
WORD nWhiteValue,
WORD nExposeAttempts,
WORD nGap,
BOOL Invert
void (*fpProgress) (WORD)
)

Return Values

RESULT_SUCCESS
RESULT_ERR_UNSUPPORTED
RESULT_ERR_DRIVER
RESULT_ERR_PARAMETER
RESULT_ERR_NORESPONSE
RESULT_ERR_BADREGION
RESULT_ERR_MEMORY
RESULT_ERR_FILE

Parameters

pImageBuffer

Memory pointer to where the image should be stored. The caller must allocate this buffer before calling oemGetImage.

pdwSize

Upon successful return, the number of bytes of image data stored in pImageBuffer.

nTop, nLeft

Coordinates relative to the image engine's pixel grid for first pixel of the transferred image. The upper left pixel has both an nLeft and an nTop value of 0.

nRight, nBottom

Coordinates relative to the image engine's pixel grid for last pixel of the transferred image.

nSkip:

When transferring an image, transfer every nSkip pixel.

nBits

The color depth for the transferred image. Valid values are typically only 8 or 1, but this depends on the engine hardware. Independent of the bits used, a lower value is darker than a higher value. For example, if 8 bits are chosen, then a pixel value of 255 indicates pure white, and a pixel value of 0 indicates pure black. Values in between are to be interpreted as incremental levels of gray.

nFormat: *FF_RAW_BINARY*

The black and white data stored in the pImageBuffer is stored 1 bit per pixel starting with the upper left pixel and proceeding sequentially left to right and top to bottom.

FF_RAW_GRAY

The grayscale data stored in the pImageBuffer is stored 8 bit per pixel starting with the upper left pixel and proceeding sequentially left to right and top to bottom.

nWhiteValue

The target white value when performing auto exposure control.

nExposeAttempts

The number of attempts the unit makes to get the image to the correct exposure level.

nGap

How close the white value of the image must be to the nWhiteValue for the image to be accepted. A value of 0 (zero) can be passed in to cause the unit to use its pre-defined value. For example, if you want to use 10 for the nGap, and the default value for nWhiteValue, then pass in 10 for nGap, and 0 for nWhite.

Invert

The image is rotated 180° (upside down). This allows you to invert images for platforms where the imager is mounted upside down.

fpProgress

Reserved. Must be NULL.

oemGetImageData

This function gets data from the Imager. Each subsequent call fills the supplied buffer with image data. The end of the image data is identified by a GetImageData() call that does not fill the buffer (less than BufferSize bytes were placed in the buffer) and has a return of RESULT_SUCCESS.

Prototype

```
Result_t    oemGetImageData  (
            DWORD dwNumberToRead,
            BYTE *pBuffer,
            DWORD *pNumBytesRead
            )
```

Return Values

RESULT_SUCCESS
RESULT_ERR_DRIVER
RESULT_ERR_PARAMETER

Parameters

dwNumberToRead

Size of the user supplied buffer to be filled with image data.

pBuffer

Pointer to the user supplied buffer that holds the image data.

pNumBytesRead:

Returns the number of bytes of image data read.

oemGetImagerInfo

This function returns information about imaging capability of the connected device.

```
Result_t    oemGetImagerInfo (
            WORD *pnCols,
            WORD *pnRows,
            WORD *pnBits
            )
```

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

pnCols

Upon successful return, this variable points to the number of column pixels in the Imager.

pnRows

Upon successful return, this variable points to the number of row pixels in the Imager.

pnBits

Upon successful return, this variable points to the number of bits per pixel supported by the Imager.

oemGetImagerProperties

This function returns information about the imager.

Result_t oemGetImagerProperties (
ImagerProperties_t *pImgProp
)

Return Values

RESULT_ERR_PARAMETER
RESULT_ERR_SUCCESS

Parameters

pImgProp

Pointer to structure to be filled by this function with information about the imager.

```
typedef struct
{
   DWORD dwSize;
   DWORD dwEngineID;// 0
   DWORD dwImagerRows;
   DWORD dwImagerCols;
   DWORD dwBitsPerPixel;
   DWORD dwRotation;
   DWORD dwAimerXoffset;
   DWORD dwAimerYoffset;
   DWORD dwYDepth;
      } ImagerProperties_t;
```

dwSize: Size of structure

dwEngineID: EngineIDs values

TYPE_NONE =0 (No imager hardware)

TYPE_IT4200 =1

TYPE_IT4000 =5

TYPE_IT4100 =6

TYPE_IT4300 =7

dwImagerRows: Number of rows for a given imager

dwImagerCols: Number of columns for a given imager.

dwBitsPerPixel: Typically this is 8 for byte pixels.

dwRotation: RIGHT_SIDE_UP = 0
ROTATED_RIGHT
UPSIDE_DOWN
ROTATED_LEFT

dwAimerXoffset: This value represents the X coordinate for the center of the aimer pattern.

dwAimerYoffset: This value represents the Y coordinate for the center of the aimer pattern.

oemGetLastImage

This function is used to retrieve the last image acquired by the image engine.

```
Result_t    oemGetLastImage (
            BYTE *pImageBuffer
            DWORD *pImageSize
            )
```

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER
RESULT_ERR_NOIMAGE

Parameters

pImageBuffer

Pointer to the user supplied buffer to be filled with the image data. The buffer must be at least MAX_IMAGE_SIZE in size.

pImageSize

Pointer to a DWORD that returns the size of the image.

oemGetLastImageExt

This function is used to retrieve the last image acquired by the image engine and the exposure parameters associated with that image.

```
Result_t    oemGetLastImageExt (
            BYTE  *pImageBuffer,
            DWORD *pImageSize,
            ImageAttributes_t  *pImageAttributes
            )
```

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER
RESULT_ERR_NOIMAGE

Parameters

pImageBuffer

Pointer to the user-supplied buffer to be filled with the image data. The buffer must be at least MAX_IMAGE_SIZE in size.

pImageSize

Pointer to a DWORD that will return the size of the image.

pImageAttributes

Pointer to an ImageAttributes_t structure that has been initialized to zeros. Upon successful return the structure will contain the specific values for the returned image. Please see the engnapi.h header file for details of the ImageAttributes_t structure.

oemGetLastImageSize

This function returns the number of rows, number of columns and size in bytes of the last image returned by one of the other imaging functions: oemAcquireImage, oemGetImage/oemGetImageData or oemImageStreamRead.

Result_t ***oemGetLastImageSize (***
WORD *pwCols,
WORD *pwRows,
DWORD *pdwSize
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

pwCols

Upon successful return, this variable points to the number of column pixels in the Image.

pwRows

Upon successful return, this variable points to the number of row pixels in the Image.

pdwBytes

Upon successful return, this variable points to the total number of Image bytes.

oemGetLeaveLightsOn

This function returns a parameter reflecting the operational mode of the illumination LEDs during scanning.

Result_t ***oemGetLeaveLightsOn (***
SetupType_t SetupType,
BOOL *pbEnable
)

Return Values

RESULT_SUCCESS
RESULT_ERR_NOTCONNECTED
RESULT_ERR_UNSUPPORTED
RESULT_ERR_DRIVER
RESULT_ERR_PARAMETER
RESULT_ERR_NORESPONSE

Parameters

SetupType

SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

pbEnable

Upon successful return, this variable reflects the operational mode of the illumination LEDs during scanning. If FALSE, the illumination LEDs will be in normal operational mode and will be flashed on and off during scanning. If TRUE, the illumination LEDs are always on during scanning and will not flash.

oemGetLinearRange

This function is used to get the size of the window used in the Advanced Linear decoding mode.

Result_t ***oemGetLinearRange (***
SetupType_t SetupType,
WORD *pnLinearRange
)

Return Values

RESULT_SUCCESS
RESULT_ERR_UNSUPPORTED

Parameters

SetupType
SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

pnLinearRange
Points to a WORD variable where the range value will be placed.

oemGetMaxMessageChars

This function returns the size of the largest possible decode message in characters.

Result_t ***oemGetMaxMessageChars (***
WORD *pnChars
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

pnChars
Upon successful return, this variable points to the number of characters (not bytes) required for the largest possible decode message. In WindowsCE the character size is 2 bytes (wchar_t)

oemGetPrintWeight

This function returns the current or default "Print Weight" (relative contrast) expected by the decoder for barcodes or OCR text.

Result_t ***oemGetPrintWeight (***
SetupType_t SetupType,
WORD *pnPrintWeight
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

SetupType
SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

pnPrintWeight
Upon successful return, this variable points to the print weight (relative blackness) that the decoder expects to see when attempting to decode symbols or OCR text from an image.

oemGetScanDriverRevision

This function returns the revision of the scan driver component that interfaces to the OEM API layer.

Result_t ***oemGetScanDriverRevision(***
TCHAR *pszRev
)

Return Values

RESULT_ERR_PARAMETER
RESULT_ERR_UNSUPPORTED
RESULT_ERR_SUCCESS

Parameters

pszRev

Pointer to string that will contain the revision string. This should be allocated prior to the call and contain 80 characters.

oemGetSearchTimeLimit

This function is used to retrieve the current search maximum time limit. The limit, specified in milliseconds, is the maximum amount of time the search process may use to look for potential labels in the current image.

Result_t ***oemGetSearchTimeLimit (***
SetupType_t SetupType,
WORD *pnLimit
)

Return Values

RESULT_SUCCESS
RESULT_ERR_UNSUPPORTED
RESULT_ERR_PARAMETER

Parameters

Setup Type

SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

pnLimit

Points to a word variable that upon return of RESULT_SUCCESS will contain the maximum search time limit. A value of zero indicates no limit.

oemGetSetupAll

This function is used to get all symbology options.

Result_t ***oemGetSetupAll (***
SetupType_t SetupType,
BOOL *pEnabled
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

SetupType

SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

pEnabled

Points to an array of symbologies with a size of MAX_SYMBOLOGIES. This array should be declared and cleared. Upon return, this array will contain the enable/disable status of each symbology.

oemGetSetupAusPost

This function is used to get the Australian Postal Code symbology-specific options.

Result_t ***oemGetSetupAusPost (***
SetupType_t SetupType,
BOOL *pEnabled
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

SetupType

SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

pEnabled

Points to a BOOL variable that contains the enabled state of this symbology upon returning from the function. A TRUE means the symbology is enabled; a FALSE means the symbology is disabled.

oemGetSetupAztec

This function is used to get the Aztec and Aztec Mesa Code symbology-specific options.

Result_t ***oemGetSetupAztec (***
SetupType_t SetupType,
BOOL *pEnabled,
WORD *pMinLength,
WORD *pMaxLength
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

SetupType

SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

pEnabled

Points to a BOOL variable that contains the enabled state of this symbology upon returning from the function. A TRUE means the symbology is enabled; a FALSE means the symbology is disabled.

pMinLength

Points to a WORD variable that contains the minimum length decoded Aztec or Aztec Mesa Code message the engine should return. Aztec or Aztec Mesa Code messages smaller than this minimum length are not reported by the engine. The default value is 1.

pMaxLength

Points to a WORD variable that contains the maximum length decoded Aztec or Aztec Mesa Code message the engine should return. Aztec or Aztec Mesa Codes messages larger than this maximum length are reported by the engine. The default value is 3750.

oemGetSetupBPO

This function is used to get the British Post symbology-specific options.

Result_t ***oemGetSetupBPO (***
SetupType_t SetupType,
BOOL *pEnabled
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

SetupType

SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

pEnabled

Points to a BOOL variable that contains the enabled state of this symbology upon returning from the function. A TRUE means the symbology is enabled; a FALSE means the symbology is disabled.

oemGetSetupCanPost

This function is used to get the Canadian Post symbology-specific options.

Result_t oemGetSetupCanPost (
SetupType_t SetupType,
BOOL *pEnabled
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

SetupType

SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

pEnabled

Points to a BOOL variable that contains the enabled state of this symbology upon returning from the function. A TRUE means the symbology is enabled; a FALSE means the symbology is disabled.

oemGetSetupChinaPost

This function is used to get the China Post symbology specific options.

Result_t oemGetSetupChinaPost (
SetupType_t SetupType,
BOOL *pEnabled,
WORD *pMinLength,
WORD *pMaxLength
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

SetupType

SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

pEnabled

Points to a BOOL variable that contains the enabled state of this symbology upon returning from the function. A TRUE means the symbology is enabled; a FALSE means the symbology is disabled.

pMinLength

Points to a WORD variable that contains the minimum length decoded China Post message the engine should return. China Post messages smaller than this minimum length are not reported by the engine. The default value is 4.

pMaxLength

Points to a WORD variable that contains the maximum length decoded China Post message the engine should return. China Post messages larger than this maximum length are not reported by the engine. The default value is 80.

oemGetSetupCodabar

This function is used to get the Codabar symbology-specific options.

Result_t ***oemGetSetupCodabar (***
SetupType_t SetupType,
BOOL *pEnabled,
WORD *pMinLength,
WORD *pMaxLength
BOOL *pbSSXmit,
BOOL *pbCheckCharOn,
BOOL *pbXmitCheckChar
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

SetupType

SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

pEnabled

Points to a BOOL variable that contains the enabled state of this symbology upon returning from the function. A TRUE means the symbology is enabled; a FALSE means the symbology is disabled.

pMinLength

Points to a WORD variable that contains the minimum length decoded Codabar message the engine should return. Codabar messages smaller than this minimum length are not reported by the engine. The default value is 2.

pMaxLength

Points to a WORD variable that contains the maximum length decoded Codabar message the engine should return. Codabar messages larger than this maximum length are reported by the engine. The default value is 60.

pbSSXmit

Points to a BOOL variable that determines if the start and stop characters are returned in the data string after a successful Codabar decode. If bSSXmit is TRUE, the start and stop characters are included. If FALSE, they are not included. The default value is FALSE.

pbCheckCharOn

Points to a BOOL variable that determines if the engine will read Codabar bar codes with or without check characters. If TRUE, the engine only decodes Codabar codes with a check character. If FALSE, the decoder decodes codes with or without a check character. The default value is FALSE.

pbXmitCheckChar

Points to a BOOL variable that determines if the engine will return the check character as part of the data string after a successful decode. If TRUE, the engine returns the check character. If FALSE the check character is not returned. The default value is FALSE.

oemGetSetupCodablock

This function is used to get the Codablock F symbology-specific options.

Result_t ***oemGetSetupCodablock (***
SetupType_t SetupType,
BOOL *pEnabled,
WORD *pMinLength,
WORD *pMaxLength
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

SetupType

SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

pEnabled

Points to a BOOL variable that contains the enabled state of this symbology upon returning from the function. A TRUE means the symbology is enabled; a FALSE means the symbology is disabled.

pMinLength

Points to a WORD variable that contains the minimum length decoded Codablock message the engine should return. Codablock messages smaller than this minimum length are not reported by the engine. The default value is 0.

pMaxLength

Points to a WORD variable that contains the maximum length decoded Codablock message the engine should return. Codablock messages larger than this maximum length are reported by the engine. The default value is 2048.

pbXmitCheckChar

Points to a BOOL variable that determines if the engine will return the check character as part of the data string after a successful decode. If TRUE, the engine returns the check character. If FALSE the check character is not returned. The default value is FALSE.

oemGetSetupCode11

This function is used to get the Code 11 symbology-specific options.

Result_t oemGetSetupCode11 (
SetupType_t SetupType,
BOOL *pEnabled,
WORD *pMinLength,
WORD *pMaxLength,
BOOL *pbXmitCheckChar
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

SetupType

SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

pEnabled

Points to a BOOL variable that contains the enabled state of this symbology upon returning from the function. A TRUE means the symbology is enabled; a FALSE means the symbology is disabled.

pMinLength

Points to a WORD variable that contains the minimum length decoded Code 11 message the engine should return. Code 11 messages smaller than this minimum length are not reported by the engine. The default value is 4.

pMaxLength

Points to a WORD variable that contains the maximum length decoded Code 11 message the engine should return. Code 11 messages larger than this maximum length are reported by the engine. The default value is 80.

pbXmitCheckChar

Points to a BOOL variable that determines if the engine will return the check character as part of the data string after a successful decode. If TRUE, the engine returns the check character. If FALSE the check character is not returned. The default value is FALSE.

oemGetSetupCode128

This function is used to get the Code 128 symbology-specific options.

Result_t ***oemGetSetupCode128 (***
SetupType_t SetupType,
BOOL *pEnabled,
WORD *pMinLength,
WORD *pMaxLength
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

SetupType

SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

pEnabled

Points to a BOOL variable that contains the enabled state of this symbology upon returning from the function. A TRUE means the symbology is enabled; a FALSE means the symbology is disabled.

pMinLength

Points to a WORD variable that contains the minimum length decoded Code 128 message the engine should return. Code 128 messages smaller than this minimum length are not reported by the engine. The default value is 0.

pMaxLength

Points to a WORD variable that contains the maximum length decoded Code 128 message the engine should return. Code 128 messages larger than this maximum length are reported by the engine. The default value is 80.

oemGetSetupCode16K

This function is used to get the Code 16K symbology specific options.

Result_t ***oemGetSetupCode16K (***
SetupType_t SetupType,
BOOL *pEnabled,
WORD *pMinLength,
WORD *pMaxLength
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

SetupType

SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

pEnabled

Points to a BOOL variable that contains the enabled state of this symbology upon returning from the function. A TRUE means the symbology is enabled; a FALSE means the symbology is disabled.

pMinLength

Points to a WORD variable that contains the minimum length decoded Code 16K message the engine should return. Code 16K messages smaller than this minimum length are not reported by the engine. The default value is 1.

pMaxLength

Points to a WORD variable that contains the maximum length decoded Code16K message the engine should return. Code 16K messages larger than this maximum length are not reported by the engine. The default value is 160.

oemGetSetupCode32

This function is used to get the Code 32 symbology specific options.

Result_t ***oemGetSetupCode32 (***
SetupType_t SetupType,
BOOL *pEnabled,
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

SetupType

SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

pEnabled

Points to a BOOL variable that contains the enabled state of this symbology upon returning from the function. A TRUE means the symbology is enabled; a FALSE means the symbology is disabled.

oemGetSetupCode39

This function is used to get the Code 39 symbology-specific options.

Result_t ***oemGetSetupCode39 (***
SetupType_t SetupType,
BOOL *pEnabled,
WORD *pMinLength,
WORD *pMaxLength,
BOOL *pbSSXmit,
BOOL *pbFullAscii,
BOOL *pbAppend,
BOOL *pbCheckCharOn,
BOOL *bXmitCheckChar
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

SetupType

SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

pEnabled

Points to a BOOL variable that contains the enabled state of this symbology upon returning from the function. A TRUE means the symbology is enabled; a FALSE means the symbology is disabled.

pMinLength

Points to a WORD variable that contains the minimum length decoded Code 39 message the engine should return. Code 39 messages smaller than this minimum length are not reported by the engine. The default value is 2.

pMaxLength

Points to a WORD variable that contains the maximum length decoded Code 39 message the engine should return. Code 39 messages larger than this maximum length are reported by the engine. The default value is 48.

pbSSXmit

Points to a BOOL variable that determines if the start and stop characters are returned in the data string after a successful Code 39 decode. If bSSXmit is TRUE, the start and stop characters are included. If FALSE, they are not included. The default value is FALSE.

pbFullAscii

Points to a BOOL variable that determines if certain character pairs within the bar code symbol are interpreted and returned as a single character. If bFullAscii is TRUE, interpretation is enabled. If FALSE, no interpretation is attempted. The default value is FALSE.

pbAppend

Points to a BOOL variable that determines if the engine should append together and buffer up Code 39 symbols that start with a space (excluding the start and stop characters). The engine stores the symbols in the order in which they are read. It returns the data after a Code 39 symbol with no leading space is read. The return data has the leading spaces removed. If TRUE, the append feature is enabled. If FALSE, the append feature is disabled. The default value is FALSE.

pbCheckCharOn

Points to a BOOL variable that determines if the engine will read Code 39 bar codes with or without check characters. If TRUE, the engine only decodes Code 39 codes with a check character. If FALSE, the decoder decodes codes with or without a check character. The default value is FALSE.

pbXmitCheckChar

Points to a BOOL variable that determines if the engine will return the check character as part of the data string after a successful decode. If TRUE, the engine returns the check character. If FALSE, the check character is not returned. The default value is FALSE.

oemGetSetupCode49

This function is used to get the Code 49 symbology-specific options.

```
Result_t    oemGetSetupCode49 (
            SetupType_t SetupType,
            BOOL *pEnabled,
            WORD *pMinLength,
            WORD *pMaxLength
            )
```

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

SetupType

SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

pEnabled

Points to a BOOL variable that contains the enabled state of this symbology upon returning from the function. A TRUE means the symbology is enabled; a FALSE means the symbology is disabled.

pMinLength

Points to a WORD variable that contains the minimum length decoded Code 49 message the engine should return. Code 49 messages smaller than this minimum length are not reported by the engine. The default value is 1.

pMaxLength

Points to a WORD variable that contains the maximum length decoded Code 49 message the engine should return. Code 49 messages larger than this maximum length are reported by the engine. The default value is 81.

oemGetSetupCode93

This function is used to get the Code 93 symbology-specific options.

Result_t oemGetSetupCode93 (
SetupType_t SetupType,
BOOL *pEnabled,
WORD *pMinLength,
WORD *pMaxLength
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

SetupType

SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

pEnabled

Points to a BOOL variable that contains the enabled state of this symbology upon returning from the function. A TRUE means the symbology is enabled; a FALSE means the symbology is disabled.

pMinLength

Points to a WORD variable that contains the minimum length decoded Code 93 message the engine should return. Code 93 messages smaller than this minimum length are not reported by the engine. The default value is 0.

pMaxLength

Points to a WORD variable that contains the maximum length decoded Code 93 message the engine should return. Code 93 messages larger than this maximum length are reported by the engine. The default value is 80.

oemGetSetupComposite

This function is used to get the EANoUCC Composite symbology-specific options.

Result_t oemGetSetupComposite (
SetupType_t SetupType,
BOOL *pEnabled,
WORD *pMinLength,
WORD *pMaxLength
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

SetupType

SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

pEnabled

Points to a BOOL variable that contains the enabled state of this symbology upon returning from the function. A TRUE means the symbology is enabled; a FALSE means the symbology is disabled.

pMinLength

Points to a WORD variable that contains the minimum length decoded EANoUCC Composite message the engine should return. EANoUCC Composite messages smaller than this minimum length are not reported by the engine. The default value is 1.

pMaxLength

Points to a WORD variable that contains the maximum length decoded EANoUCC Composite message the engine should return. EANoUCC Composite messages larger than this maximum length are reported by the engine. The default value is 300.

oemGetSetupCompositeEx

This function is used to get the EANoUCC Composite, as well as other Composite symbology-specific options.

Result_t oemGetSetupCompositeEx (
SetupType_t SetupType,
BOOL *pEnabled,
WORD *pMinLength,
WORD *pMaxLength,
BOOL *pbCompositeOnUpcEan
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

SetupType

SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

pEnabled

Points to a BOOL variable that contains the enabled state of this symbology upon returning from the function. A TRUE means the symbology is enabled; a FALSE means the symbology is disabled.

pMinLength

Points to a WORD variable that contains the minimum length decoded Composite message the engine should return. EANoUCC Composite messages smaller than this minimum length are not reported by the engine. The default value is 1.

pMaxLength

Points to a WORD variable that contains the maximum length decoded Composite message the engine should return. EANoUCC Composite messages larger than this maximum length are reported by the engine. The default value is 2750.

pbCompositeOnUpcEan

Points to a BOOL variable that contains the enabled state of EANoUCC Composite code associated with EAN and UPC codes.

oemGetSetupCouponCode

This function is used to get the UPC-A with extended Coupon Code symbology specific options.

Result_t ***oemGetSetupCouponCode (***
SetupType_t SetupType,
BOOL *pEnabled
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

SetupType

SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

pEnabled

Points to a BOOL variable that contains the enabled state of this symbology upon returning from the function. A TRUE means the symbology is enabled; a FALSE means the symbology is disabled.

oemGetSetupDataMatrix

This function is used to get the Data Matrix symbology-specific options.

Result_t ***oemGetSetupDataMatrix (***
SetupType_t SetupType,
BOOL *pEnabled,
WORD *pMinLength,
WORD *pMaxLength
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

SetupType

SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

pEnabled

Points to a BOOL variable that contains the enabled state of this symbology upon returning from the function. A TRUE means the symbology is enabled; a FALSE means the symbology is disabled.

pMinLength

Points to a WORD variable that contains the minimum length decoded Data Matrix message the engine should return. Data Matrix messages smaller than this minimum length are not reported by the engine. The default value is 1.

pMaxLength

Points to a WORD variable that contains the maximum length decoded Data Matrix message the engine should return. Data Matrix messages larger than this maximum length are reported by the engine. The default value is 1500.

oemGetSetupDutchPost

This function is used to get theKIX (Netherlands) Post symbology-specific options.

Result_t ***oemGetSetupDutchPost (***
SetupType_t SetupType,
BOOL *pEnabled
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

SetupType

SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

pEnabled

Points to a BOOL variable that contains the enabled state of this symbology upon returning from the function. A TRUE means the symbology is enabled; a FALSE means the symbology is disabled.

oemGetSetupEAN8

This function is used to get the EAN-8 symbology-specific options.

Result_t ***oemGetSetupEAN8 (***
SetupType_t SetupType,
BOOL *pEnabled,
BOOL *bXmitCheckChar,
BOOL *pAddendaReq,
BOOL *bAddendaSeparator,
BOOL *bAddenda2Digit,
BOOL *bAddenda5Digit
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

SetupType

SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

pEnabled

Points to a BOOL variable that contains the enabled state of this symbology upon returning from the function. A TRUE means the symbology is enabled; a FALSE means the symbology is disabled.

bXmitCheckChar

Points to a BOOL variable that determines if the engine will return the check character as part of the data string after a successful decode. If TRUE, the engine returns the check character. If FALSE, the check character is not returned. The default value is FALSE.

pAddendaReq

Points to a BOOL variable that determines if the engine will decode only EAN bar codes that have a 2 or 5 digit addenda. If TRUE, the engine decodes only EAN symbols with an addenda. If FALSE, the engine decodes all enabled EAN symbols. The default value is FALSE.

bAddendaSeparator

Points to a BOOL variable that determines if there is a space character between the data from the bar code and the data from the addenda. If TRUE, there is a space. If FALSE, there is no space. The default value is FALSE.

bAddenda2Digit

Points to a BOOL variable that determines if the engine will look for a 2 digit addenda at the end of the EAN bar code. If TRUE, and an addenda is present, the engine adds the two digit addenda data to the end of the message. If FALSE, the engine ignores addenda data. The default value is FALSE.

bAddenda5Digit

Points to a BOOL variable that determines if the engine will look for a 5 digit addenda at the end of the EAN bar code. If TRUE, and an addenda is present, the engine adds the five digit addenda data to the end of the message. If FALSE, the engine ignores addenda data. The default value is FALSE.

oemGetSetupEAN13

This function is used to get the EAN-13 symbology-specific options.

```
Result_t    oemGetSetupEAN8 (
            SetupType_t SetupType,
            BOOL *pEnabled,
            BOOL *bXmitCheckChar,
            BOOL *pAddendaReq,
            BOOL *bAddendaSeparator,
            BOOL *bAddenda2Digit,
            BOOL *bAddenda5Digit
            )
```

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

SetupType

SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

pEnabled

Points to a BOOL variable that contains the enabled state of this symbology upon returning from the function. A TRUE means the symbology is enabled; a FALSE means the symbology is disabled.

bXmitCheckChar

Points to a BOOL variable that determines if the engine will return the check character as part of the data string after a successful decode. If TRUE, the engine returns the check character. If FALSE, the check character is not returned. The default value is FALSE.

pAddendaReq

Points to a BOOL variable that determines if the engine will decode only EAN bar codes that have a 2 or 5 digit addenda. If TRUE, the engine decodes only EAN symbols with an addenda. If FALSE, the engine decodes all enabled EAN symbols. The default value is FALSE.

bAddendaSeparator

Points to a BOOL variable that determines if there is a space character between the data from the bar code and the data from the addenda. If TRUE, there is a space. If FALSE, there is no space. The default value is FALSE.

bAddenda2Digit

Points to a BOOL variable that determines if the engine will look for a 2 digit addenda at the end of the EAN bar code. If TRUE, and an addenda is present, the engine adds the two digit addenda data to the end of the message. If FALSE, the engine ignores addenda data. The default value is FALSE.

bAddenda5Digit

Points to a BOOL variable that determines if the engine will look for a 5 digit addenda at the end of the EAN bar code. If TRUE, and an addenda is present, the engine adds the five digit addenda data to the end of the message. If FALSE, the engine ignores addenda data. The default value is FALSE.

oemGetSetupIATA25

This function is used to get the Straight 2 of 5 IATA symbology-specific options.

```
Result_t    oemGetSetupIATA25 (
            SetupType_t SetupType,
            BOOL *pEnabled,
            WORD *pMinLength,
            WORD *pMaxLength
            )
```

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

SetupType

SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

pEnabled

Points to a BOOL variable that contains the enabled state of this symbology upon returning from the function. A TRUE means the symbology is enabled; a FALSE means the symbology is disabled.

pMinLength

Points to a WORD variable that contains the minimum length decoded IATA 2 of 5 message the engine should return. IATA 2 of 5 messages smaller than this minimum length are not reported by the engine. The default value is 4.

pMaxLength

Points to a WORD variable that contains the maximum length decoded IATA 2 of 5 message the engine should return. IATA 2 of 5 messages larger than this maximum length are reported by the engine. The default value is 80.

oemGetSetupImager

This function gets the current Imager setup values.

Prototype

Result_t ***oemGetSetupImager (***
SetupType_t SetupType,
ImagerSetup_t *pImagerSetup
)

Return Values

RESULT_SUCCESS

Parameters

pImagerSetup
Data structure that sets up the Imager. See oemAcquireImage (page 2-1) for details.

SetupType
This parameter is ignored.

oemGetSetupInt25

This function is used to get the Interleaved 2 of 5 symbology-specific options.

Result_t ***oemGetSetupInt25 (***
SetupType_t SetupType,
BOOL *pEnabled,
WORD *pMinLength,
WORD *pMaxLength,
BOOL *pbCheckDigitOn,
BOOL *pbXmitCheckDigit
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

SetupType
SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

pEnabled
Points to a BOOL variable that contains the enabled state of this symbology upon returning from the function. A TRUE means the symbology is enabled; a FALSE means the symbology is disabled.

pMinLength
Points to a WORD variable that contains the minimum length decoded Interleaved 2 of 5 message the engine should return. Interleaved 2 of 5 messages smaller than this minimum length are not reported by the engine. The default value is 4.

pMaxLength
Points to a WORD variable that contains the maximum length decoded Interleaved 2 of 5 message the engine should return. Interleaved 2 of 5 messages larger than this maximum length are reported by the engine. The default value is 80.

pbCheckDigitOn
Points to a BOOL variable that determines if the engine will read Interleaved 2 of 5 bar codes with or without check characters. If TRUE, the engine only decodes Interleaved 2 of 5 codes with a check digit. If FALSE, the decoder decodes codes with or without a check digit. The default value is FALSE.

pbXmitCheckDigit
Points to a BOOL variable that determines if the engine will return the check digit as part of the data string after a successful decode. If TRUE, the engine returns the check digit. If FALSE, the check digit is not returned. The default value is FALSE.

oemGetSetupISBT

This function is used to get the ISBT 128 symbology-specific options.

Result_t ***oemGetSetupISBT (***
SetupType_t SetupType,
BOOL *pEnabled
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

SetupType

SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

pEnabled

Points to a BOOL variable that contains the enabled state of this symbology upon returning from the function. A TRUE means the symbology is enabled; a FALSE means the symbology is disabled.

oemGetSetupJapost

This function is used to get the Japanese Post symbology-specific options.

Result_t ***oemGetSetupJapost (***
SetupType_t SetupType,
BOOL *pEnabled
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

SetupType

SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

pEnabled

Points to a BOOL variable that contains the enabled state of this symbology upon returning from the function. A TRUE means the symbology is enabled; a FALSE means the symbology is disabled.

oemGetSetupKoreanPost

This function is used to get the Korean Post symbology specific options.

Result_t ***oemGetSetupKoreanPost (***
SetupType_t SetupType,
BOOL *pEnabled,
WORD *pMinLength,
WORD *pMaxLength
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

SetupType

SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

pEnabled

Points to a BOOL variable that contains the enabled state of this symbology upon returning from the function. A TRUE means the symbology is enabled; a FALSE means the symbology is disabled.

pMinLength

Points to a WORD variable that contains the minimum length decoded Korean Post message the engine should return. Korean Post messages smaller than this minimum length are not reported by the engine. The default value is 4.

pMaxLength

Points to a WORD variable that contains the maximum length decoded Korean Post message the engine should return. Korean Post messages larger than this maximum length are reported by the engine. The default value is 48.

oemGetSetupMaxicode

This function is used to get the MaxiCode symbology-specific options.

```
Result_t    oemGetSetupMaxicode (
            SetupType_t SetupType,
            BOOL *pEnabled,
            WORD *pMinLength,
            WORD *pMaxLength,
            BOOL *pCarrierMsgOnly
            )
```

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

SetupType

SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

pEnabled

Points to a BOOL variable that contains the enabled state of this symbology upon returning from the function. A TRUE means the symbology is enabled; a FALSE means the symbology is disabled.

pMinLength

Points to a WORD variable that contains the minimum length decoded Maxicode message the engine should return. Maxicode messages smaller than this minimum length are not reported by the engine. The default value is 1.

pMaxLength

Points to a WORD variable that contains the maximum length decoded Maxicode message the engine should return. Maxicode messages larger than this maximum length are reported by the engine. The default value is 150.

pCarrierMsgOnly

Points to a BOOL variable that determines if the engine will return only the Structured Carrier Message portion of the decoded message. When TRUE, the engine only returns the Structured Carrier Message data. When FALSE, the engine returns the entire message. The default value is FALSE.

oemGetSetupMesa

This function is used to get the Aztec Mesa Code symbology-specific options.

Result_t ***oemGetSetupMesa (***
SetupType_t SetupType,
BOOL *pUMSEnabled,
BOOL *pEMSEnabled,
BOOL *p3MSEnabled,
BOOL *p1MSEnabled,
BOOL *pIMSEnabled,
BOOL *p9MSEnabled
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

SetupType
SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

pUMSEnabled
Points to a BOOL variable that contains the enabled state of UPCA Mesa. TRUE = Enabled, FALSE = Disabled.

pEMSEnabled
Points to a BOOL variable that contains the enabled state of EAN13 Mesa. TRUE = Enabled, FALSE = Disabled.

p3MSEnabled
Points to a BOOL variable that contains the enabled state of Code 39 Mesa. TRUE = Enabled, FALSE = Disabled.

p1MSEnabled
Points to a BOOL variable that contains the enabled state of Code 128 Mesa. TRUE = Enabled, FALSE = Disabled.

pIMSEnabled
Points to a BOOL variable that contains the enabled state of Interleaved 2 of 5 Mesa. TRUE = Enabled, FALSE = Disabled.

p9MSEnabled
Points to a BOOL variable that contains the enabled state of Code 93 Mesa.

oemGetSetupMicroPDF

This function is used to get the MicroPDF417 symbology-specific options.

Result_t ***oemGetSetupMicroPDF (***
SetupType_t SetupType,
BOOL *pEnabled,
WORD *pMinLength,
WORD *pMaxLength
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

SetupType

SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

pEnabled

Points to a BOOL variable that contains the enabled state of this symbology upon returning from the function. A TRUE means the symbology is enabled; a FALSE means the symbology is disabled.

pMinLength

Points to a WORD variable that contains the minimum length decoded MicroPDF417 message the engine should return. MicroPDF417 messages smaller than this minimum length are not reported by the engine. The default value is 1.

pMaxLength

Points to a WORD variable that contains the maximum length decoded MicroPDF417 message the engine should return. MicroPDF417 messages larger than this maximum length are reported by the engine. The default value is 2750.

oemGetSetupMSI

This function is used to get the MSI symbology-specific options.

Result_t oemGetSetupMSI (
SetupType_t SetupType,
BOOL *pEnabled,
WORD *pMinLength,
WORD *pMaxLength,
BOOL *pbXmitCheckChar
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

SetupType

SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

pEnabled

Points to a BOOL variable that contains the enabled state of this symbology upon returning from the function. A TRUE means the symbology is enabled; a FALSE means the symbology is disabled.

pMinLength

Points to a WORD variable that contains the minimum length decoded MSI message the engine should return. MSI messages smaller than this minimum length are not reported by the engine. The default value is 4.

pMaxLength

Points to a WORD variable that contains the maximum length decoded MSI message the engine should return. MSI messages larger than this maximum length are reported by the engine. The default value is 48.

pbXmitCheckChar

Points to a BOOL variable that determines if the engine will return the check character as part of the data string after a successful decode. If TRUE, the engine returns the check character. If FALSE, the check character is not returned. The default value is FALSE.

oemGetSetupMx25

This function is used to get the Matrix 2 of 5 symbology specific options.

Result_t ***oemGetSetupMx25 (***
SetupType_t SetupType,
BOOL *pEnabled,
WORD *pMinLength,
WORD *pMaxLength
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

SetupType

SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

pEnabled

Points to a BOOL variable that contains the enabled state of this symbology upon returning from the function. A TRUE means the symbology is enabled; a FALSE means the symbology is disabled.

pMinLength

Points to a WORD variable that contains the minimum length decoded Matrix 2 of 5 message the engine should return. Matrix 2 of 5 messages smaller than this minimum length are not reported by the engine. The default value is 4.

pMaxLength

Points to a WORD variable that contains the maximum length decoded Matrix 2 of 5 message the engine should return. Matrix 2 of 5 messages larger than this maximum length are not reported by the engine. The default value is 80.

oemGetSetupOCR

This function is used to get the Optical Character Recognition (OCR) decoding options.

Result_t ***oemGetSetupOCR (***
SetupType_t SetupType,
OCRMode_t *nFont,
TCHAR *pszTemplate,
TCHAR *pszGroupG,
TCHAR *pszGroupH,
TCHAR *pszCheckChar,
OCRDirection_t *nDirection
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

SetupType

SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

nFont

This determines which OCR fonts (if any) are selected for decoding. The following values are used:

OCR_DISABLED
OCR_A
OCR_B
OCR_MONEY
OCR_MICR (currently unsupported)

pszTemplate

A null-terminated string that indicates one or more template patterns for the OCR decode. The following characters are allowed:

A-Z capital letters are matched as is
d - a digit from 0 - 9
a - alphanumeric character
l - alphabetic letter
g - any character specified in group G
h - any character specified in group H

pszGroupG

A null-terminated string that represents a list of characters that can be substituted for the lower-case 'g' in the template strings.

pszGroupH

A null-terminated string that represents a list of characters that can be substituted for the lower-case 'h' in the template strings.

pszCheckChar

A null-terminated string that represents a check character position in the template strings.

nDirection

Tells the OCR decoder which way the characters are usually oriented with respect to the image. The decoder still decodes any orientation, but use of this parameter can increase decoding speed.

oemGetSetupPDF417

This function is used to get the PDF417 symbology-specific options.

Result_t oemGetSetupPDF417(
SetupType_t SetupType,
BOOL *pEnabled,
WORD *pMinLength,
WORD *pMaxLength
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

SetupType

SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

pEnabled

Points to a BOOL variable that contains the enabled state of this symbology upon returning from the function. A TRUE means the symbology is enabled; a FALSE means the symbology is disabled.

pMinLength

Points to a WORD variable that contains the minimum length decoded PDF417 message the engine should return. PDF417 messages smaller than this minimum length are not reported by the engine. The default value is 1.

pMaxLength

Points to a WORD variable that contains the maximum length decoded PDF417 message the engine should return. PDF417 messages larger than this maximum length are reported by the engine. The default value is 2750.

oemGetSetupPlanet

This function is used to get the Planet Code symbology-specific options.

```
Result_t    oemGetSetupPlanet (
            SetupType_t SetupType,
            BOOL *pEnabled,
            BOOL *bXmitCheckDigit
            )
```

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

SetupType

SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

pEnabled

Points to a BOOL variable that contains the enabled state of this symbology upon returning from the function. A TRUE means the symbology is enabled; a FALSE means the symbology is disabled.

bXmitCheckDigit

Points to a BOOL variable that determines if the engine will return the check digit as part of the data string after a successful decode. If TRUE, the engine returns the check digit. If FALSE, the check digit is not returned. The default value is FALSE.

oemGetSetupPlessey

This function is used to get the Plessey Code symbology specific options.

```
Result_t    oemGetSetupPlessey (
            SetupType_t SetupType,
            BOOL *pEnabled,
            WORD *pMinLength,
            WORD *pMaxLength
            )
```

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

SetupType

SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

pEnabled

Points to a BOOL variable that contains the enabled state of this symbology upon returning from the function. A TRUE means the symbology is enabled; a FALSE means the symbology is disabled.

pMinLength

Points to a WORD variable that contains the minimum length decoded Plessey Code message the engine should return. Plessey Code messages smaller than this minimum length are not reported by the engine. The default value is 4.

pMaxLength

Points to a WORD variable that contains the maximum length decoded Plessey Code message the engine should return. Plessey Code messages larger than this maximum length are not reported by the engine. The default value is 48.

oemGetSetupPosiCode

This function is used to get the PosiCode symbology specific options.

```
Result_t    oemGetSetupPosiCode (
            SetupType_t SetupType,
            BOOL *pEnabled,
            WORD *pMinLength,
            WORD *pMaxLength,
            WORD *pLimited
            )
```

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

SetupType

SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

pEnabled

Points to a BOOL variable that contains the enabled state of this symbology upon returning from the function. A TRUE means the symbology is enabled; a FALSE means the symbology is disabled.

pMinLength

Points to a WORD variable that contains the minimum length decoded PosiCode message the engine should return. PosiCode messages smaller than this minimum length are not reported by the engine. The default value is 4.

pMaxLength

Points to a WORD variable that contains the maximum length decoded PosiCode message the engine should return. PosiCode messages larger than this maximum length are not reported by the engine. The default value is 48.

pLimited

Points to a WORD variable that reflects if Posicode Limited A or Posicode Limited B decoding is enabled. A value of 1 indicates Posicode Limited A is enabled , and a value of 2 indicates Posicode Limited B decoding is enabled. A value of 0 indicates that decoding of both Limited A and Limited B is disabled. The default value is 0.

oemGetSetupPostnet

This function is used to get the Postnet symbology-specific options.

Result_t ***oemGetSetupPostnet (***
SetupType_t SetupType,
BOOL *pEnabled,
BOOL *bXmitCheckDigit
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

SetupType

SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

pEnabled

Points to a BOOL variable that contains the enabled state of this symbology upon returning from the function. A TRUE means the symbology is enabled; a FALSE means the symbology is disabled.

bXmitCheckDigit

Points to a BOOL variable that determines if the engine will return the check digit as part of the data string after a successful decode. If TRUE, the engine returns the check digit. If FALSE, the check digit is not returned. The default value is FALSE.

oemGetSetupQR

This function is used to get the QR Code symbology-specific options.

Result_t ***oemGetSetupQR (***
SetupType_t SetupType,
BOOL *pEnabled,
WORD *pMinLength,
WORD *pMaxLength
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

SetupType

SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

pEnabled

Points to a BOOL variable that contains the enabled state of this symbology upon returning from the function. A TRUE means the symbology is enabled; a FALSE means the symbology is disabled.

pMinLength

Points to a WORD variable that contains the minimum length decoded QR Code message the engine should return. QR Code messages smaller than this minimum length are not reported by the engine. The default value is 1.

pMaxLength

Points to a WORD variable that contains the maximum length decoded QR Code message the engine should return. QR Code messages larger than this maximum length are reported by the engine. The default value is 3500.

oemGetSetupRSS

This function is used to get the RSS Expanded symbology-specific options.

Result_t ***oemGetSetupRSS (***
SetupType_t SetupType,
BOOL *pEnabled,
WORD *pMinLength,
WORD *pMaxLength
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

SetupType

SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

pEnabled

Points to a BOOL variable that contains the enabled state of this symbology upon returning from the function. A TRUE means the symbology is enabled; a FALSE means the symbology is disabled.

pMinLength

Points to a WORD variable that contains the minimum length decoded RSS message the engine should return. RSS messages smaller than this minimum length are not reported by the engine. The default value is 1.

pMaxLength

Points to a WORD variable that contains the maximum length decoded RSS message the engine should return. RSS messages larger than this maximum length are reported by the engine. The default value is 80.

oemGetSetupStrt25

This function is used to get the Straight 2 of 5 symbology specific options.

Result_t ***oemGetSetupStrt25 (***
SetupType_t SetupType,
BOOL *pEnabled,
WORD *pMinLength,
WORD *pMaxLength
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

SetupType

SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

pEnabled

Points to a BOOL variable that contains the enabled state of this symbology upon returning from the function. A TRUE means the symbology is enabled; a FALSE means the symbology is disabled.

pMinLength

Points to a WORD variable that contains the minimum length decoded Straight 2 of 5 message the engine should return. Straight 2 of 5 messages smaller than this minimum length are not reported by the engine. The default value is 4.

pMaxLength

Points to a WORD variable that contains the maximum length decoded Straight 2 of 5 message the engine should return. Straight 2 of 5 messages larger than this maximum length are not reported by the engine. The default value is 48.

oemGetSetupTelepen

This function is used to get the Telepen symbology specific options.

Result_t ***oemGetSetupTelepen (***
SetupType_t SetupType,
BOOL *pEnabled,
WORD *pMinLength,
WORD *pMaxLength,
BOOL *pOldStyle
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

SetupType

SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

pEnabled

Points to a BOOL variable that contains the enabled state of this symbology upon returning from the function. A TRUE means the symbology is enabled; a FALSE means the symbology is disabled.

pMinLength

Points to a WORD variable that contains the minimum length decoded Telepen message the engine should return. Telepen messages smaller than this minimum length are not reported by the engine. The default value is 1.

pMaxLength

Points to a WORD variable that contains the maximum length decoded Telepen message the engine should return. Telepen messages larger than this maximum length are not reported by the engine. The default value is 60.

pOriginal

Points to a BOOL variable that reflects if the engine is configured to reads Telepen labels that were encoded with either the original or the AIM specification. The default is FALSE.

oemGetSetupTLC39

This function is used to get the TLC39 symbology specific options.

Result_t ***oemGetSetupTLC39 (***
SetupType_t SetupType,
BOOL *pEnabled
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

SetupType

SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

pEnabled

Points to a BOOL variable that contains the enabled state of this symbology upon returning from the function. A TRUE means the symbology is enabled; a FALSE means the symbology is disabled.

oemGetSetupTrioptic

This function is used to get the Trioptic Code symbology specific options.

Result_t ***oemGetSetupTrioptic (***
SetupType_t SetupType,
BOOL *pEnabled,
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

SetupType

SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

pEnabled

Points to a BOOL variable that contains the enabled state of this symbology upon returning from the function. A TRUE means the symbology is enabled; a FALSE means the symbology is disabled.

oemGetSetupUPCA

This function is used to get the UPC-A symbology-specific options.

Result_t ***oemGetSetupUPCA (***
SetupType_t SetupType,
BOOL *pEnabled,
BOOL *pbXmitCheckChar,
BOOL *pbAddendaReq,
BOOL *pbAddendaSeparator,
BOOL *pbAddenda2Digit,
BOOL *pbAddenda5Digit,
BOOL *pbXmitNumSys
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

SetupType

SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

pEnabled

Points to a BOOL variable that contains the enabled state of this symbology upon returning from the function. A TRUE means the symbology is enabled; a FALSE means the symbology is disabled.

pbXmitCheckDigit

Points to a BOOL variable that determines if the engine will return the check digit as part of the data string after a successful decode. If TRUE, the engine returns the check digit. If FALSE, the check digit is not returned. The default value is FALSE.

pbAddendaReq

Points to a BOOL variable that determines if the engine will decode only UPC bar codes that have a 2 or 5 digit addenda. If TRUE, the engine decodes only UPC symbols with an addenda. If FALSE, the engine decodes all enabled UPC symbols. The default value is FALSE.

pbAddendaSeparator

Points to a BOOL variable that determines if there is a space character between the data from the bar code and the data from the addenda. If TRUE, there is a space. If FALSE, there is no space. The default value is FALSE.

pbAddenda2Digit

Points to a BOOL variable that determines if the engine will look for a 2 digit addenda at the end of the UPC bar code. If TRUE, and an addenda is present, the engine adds the two digit addenda data to the end of the message. If FALSE, the engine ignores addenda data. The default value is FALSE.

pbAddenda5Digit

Points to a BOOL variable that determines if the engine will look for a 5 digit addenda at the end of the UPC bar code. If TRUE, and an addenda is present, the engine adds the five digit addenda data to the end of the message. If FALSE, the engine ignores addenda data. The default value is FALSE.

pbXmitNumSys

Points to a BOOL variable that determines if the engine will return the numeric system digit of the UPC label. If TRUE, the engine returns the number system digit. If FALSE, the number system digit is not returned. The default value is TRUE.

oemGetSetupUPCE

This function is used to get the UPC-E symbology-specific options.

```
Result_t    oemGetSetupUPCE (
            SetupType_t SetupType,
            BOOL *pE0Enabled,
            BOOL *pE1Enabled,
            BOOL *pbXmitCheckDigit,
            BOOL *pbAddendaReq,
            BOOL *pbExpandVersionE,
            BOOL *pbAddendaSeparator,
            BOOL *pbAddenda2Digit,
            BOOL *pbAddenda5Digit,
            BOOL *pbXmitNumSys
            )
```

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

SetupType

SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

pE0Enabled

Points to a BOOL variable that contains the enabled state of UPC-E0 symbology upon returning from the function. A TRUE means the symbology is enabled; a FALSE means the symbology is disabled.

pE1Enabled

Points to a BOOL variable that contains the enabled state of UPC-E1 symbology upon returning from the function. A TRUE means the symbology is enabled; a FALSE means the symbology is disabled.

pbXmitCheckDigit

Points to a BOOL variable that determines if the engine will return the check digit as part of the data string after a successful decode. If TRUE, the engine returns the check digit. If FALSE, the check digit is not returned. The default value is FALSE.

pbAddendaReq

Points to a BOOL variable that determines if the engine will decode only UPC bar codes that have a 2 or 5 digit addenda. If TRUE, the engine decodes only UPC symbols with an addenda. If FALSE, the engine decodes all enabled UPC symbols. The default value is FALSE.

pbExpandVersionE

Points to a BOOL variable that determines if the engine will expand UPC-E codes to the 12 digit UPC-A format after a successful decode. If TRUE, the engine expands the code. If FALSE, the engine does not expand the UPC-E code. The default value is FALSE.

pbAddendaSeparator

Points to a BOOL variable that determines if there is a space character between the data from the bar code and the data from the addenda. If TRUE, there is a space. If FALSE, there is no space. The default value is FALSE.

pbAddenda2Digit

Points to a BOOL variable that determines if the engine will look for a 2 digit addenda at the end of the UPC bar code. If TRUE, and an addenda is present, the engine adds the two digit addenda data to the end of the message. If FALSE, the engine ignores addenda data. The default value is FALSE.

pbAddenda5Digit

Points to a BOOL variable that determines if the engine will look for a 5 digit addenda at the end of the UPC bar code. If TRUE, and an addenda is present, the engine adds the five digit addenda data to the end of the message. If FALSE, the engine ignores addenda data. The default value is FALSE.

pbXmitNumSys

Points to a BOOL variable that determines if the engine will return the numeric system digit of the UPC label. If TRUE, the engine returns the number system digit. If FALSE, the number system digit is not returned. The default value is FALSE.

oemGetVideoReverse

This function is used to determine if the decoding of inverted symbols is enabled or disabled. An inverted symbol has white bars on a black background.

```
Result_t    oemGetVideoReverse  (
            SetupType_t SetupType,
            BOOL *pbEnabled
            )
```

Return Values

RESULT_SUCCESS
RESULT_ERR_UNSUPPORTED

Parameters

Setup Type

SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

pbEnabled

A pointer to a BOOL variable that will be set to TRUE if decoding of inverted symbols is enabled and FALSE if decoding of inverted symbols is disabled.

oemImageStreamInit

This function is used to intialize the image stream interface. The caller is allowed to specifiy the skip (subsample) value.

Note: If nSkip is greater than 1, the resulting image will be subsampled by nSkip.

Result_t oemImageStreamInit (
WORD nSkip,
RECT *imgRect,
BOOL bFlip
)

Return Values

RESULT_SUCCESS
RESULT_ERR_DRIVER

Parameters

nSkip
The subsample number. A skip of 1 means all pixels, a skip of 2 means interpolate pixel pairs for every other line.

imgRect
Defines the region of the image to be returned by calls to eomImageStreamRead.

bFlip
Indicates if the returned image is to be inverted.

oemImageStreamStart

This function causes the Image Engine to start continous collecting of images. You must call oemImageStreamInit() before calling this function.

Result_t oemImageStreamStart ()

Return Values

RESULT_SUCCESS
RESULT_ERR_INITIALIZE

Parameters

None.

oemImageStreamRead

This function returns the last image acquired. The functions oemImageStreamInit() and oemImageStreamStart() must be called before calling this function. The image returned will be formatted per the parameters passed to oemImageStreamInit().

Result_t oemImageStreamRead (
BYTE *pImageBuffer,
DWORD *pdwSize
)

Return Values

RESULT_SUCCESS
RESULT_ERR_NOIMAGE

Parameters

pImageBuffer
Buffer in which the image data or image bitmap file data is returned on successful read of an image.

pdwSize
Pointer to a DWORD where the number of bytes placed in the image buffer is returned.

oemImageStreamStop

This function causes the Image Engine to stop continous collecting of images started by oemImageStreamInit() and oemImageStreamStart(). If this function is not called, the imager will continue to acquire images (illumination will stay on) which causes serious drain battery power.

Result_t oemImageStreamStop ()

Return Values

RESULT_SUCCESS

Parameters

None.

oemLeaveLightsOn

This function configures the illumination LEDs to always be on, or in normal mode during scanning.

```
Result_t    oemLeaveLightsOn (
            SetupType_t SetupType,
            BOOL  bEnable
                        )
```

Return Values

RESULT_SUCCESS
RESULT_ERR_NOTCONNECTED
RESULT_ERR_UNSUPPORTED
RESULT_ERR_DRIVER
RESULT_ERR_PARAMETER
RESULT_ERR_NORESPONSE

Parameters

SetupType

SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

bEnable

If FALSE, the illumination LEDs will be in normal operational mode and will be flashed on and off during scanning. If TRUE, the illumination LEDs are always on during scanning and will not flash.

oemLightsOn

This function turns the engine's illumination LEDs on and off.

```
Result_t    oemLightsOn  (
            BOOL bEnable
            )
```

Return Values

RESULT_SUCCESS
RESULT_ERR_DRIVER
RESULT_ERR_NORESPONSE

Parameters

bEnable

If TRUE, the illumination LEDs are turned on; otherwise they're turned off.

oemPowerOffImager

Note: This function is only available to certain applications, and is dependent upon the hardware that supplies switchable power to the imager (V).

This function allows the application to fully power down the imager for additional power control. This gives additional power control to shutdown/resume (normally the only time the sensor fully powers off), but should be balanced since there are time penalties associated with powering back up.

Result_t oemPowerOffImager(
void
)

Return Values

RESULT_SUCCESS
RESULT_ERR_UNSUPPORTED

Parameters

None.

oemSetDecodeAttemptLimit

This function is used to set the decode attempt maximum time limit. The limit, specified in milliseconds, is the maximum amount of time the decoder may use to attempt a decode on the current image.

Result_t oemSetDecodeAttemptLimit (
SetupType_t SetupType,
WORD nLimit
)

Return Values

RESULT_SUCCESS
RESULT_ERR_UNSUPPORTED
RESULT_ERR_PARAMETER

Parameters

Setup Type

SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

nLimit

The maximum amount of time the decoder may use to attempt a decode on the current image. Valid range for nLimit is 1-10,000. A value of zero indicates no limit. The default value is 0.

oemSetDecodeCenteringWindow

This function allows the caller to Enable/Setup decode centering mode. In this mode, a decode call is only successful if the area bounding the decoded symbol intersects a caller-defined rectangle located about the center of the captured image.

Note: This function allows the engine to discriminate symbols that are located physically close to each other so only one symbol is captured during decode. Only the symbol intersecting the defined rectangle is returned.

Result_t oemSetDecodeCenteringWindow (
BOOL bEnable,
RECT *pIntersectRect
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

bEnable

If TRUE, the centering is turned on; otherwise the centering is turned off.

pIntersectRect

Rectangular image region of which at least part of the decoded symbol must overlap to be considered a valid decode.

oemSetDecodeMode

This function sets the decoding mode of the engine.

Result_t ***oemSetDecodeMode (***
SetupType_t SetupType,
WORD nMode
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER
RESULT_ERR_UNSUPPORTED

Parameters

SetupType

SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

pnMode

Points to a WORD variable that contains the desired decode mode:
1 = Standard
2 = Advanced Linear
4 = Quick Omni

oemSetExposureMode

This function is used to select the exposure mode to be used during image acquisition.

Result_t ***oemSetExposureMode (***
ExposureMode_t ExposureMode
)

Return Values

RESULT_SUCCESS
RESULT_ERR_UNSUPPORTED
RESULT_ERR_DRIVER
RESULT_ERR_PARAMETER

Parameters

ExposureMode

The exposure modes that may be used are: fixed, on chip or Hand Held Products.

oemSetExposureSettings

This function is used to set various image parameters that are used during image acquisition.

Result_t ***oemSetExposureSettings (***
ExposureSettings_t *pExpsoureSettings
)

Return Values

RESULT_SUCCESS
RESULT_ERR_UNSUPPORTED
RESULT_ERR_DRIVER

Parameters

pExposureSettings

A pointer to an exposure settings structure. See the definition of the ExposureSettings_t struct in oemdecodece.h for details.

oemSetLinearRange

This function sets the size of the window used in the Advanced Linear decoding mode.

Result_t ***oemSetLinearRange (***
SetupType_t SetupType,
WORD nLinearRange
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER
RESULT_ERR_UNSUPPORTED

Parameters

SetupType

SETUP_TYPE_CURRENT for current settings, or
SETUP_TYPE_DEFAULT for default settings.

nLinearRange

Points to a WORD variable that contains the desired Range Value. 15 lines are searched for a linear bar code. Spacing between those 15 lines is determined by this value passed in. The range value is from 1 to 6. Spacing is calculated by: pixel rows to the next line = 2^(value passed in - 1)

oemSetPrintWeight

This function returns the current or default "Print Weight" (relative contrast) expected by the decoder for barcodes or OCR text.

Result_t ***oemGetPrintWeight (***
WORD nPrintWeight
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

nPrintWeight

Set the print weight (relative blackness) that the decoder expects to see when attempting to decode symbols or OCR text from an image. Changing this value can facilitate decoding of symbols with non-standard black on white contrast such as with etched metal on car parts or some soda cans.

oemSetScanningLightsMode

This function gives the user the ability to select what the illumination and aimer do during imaging.

Result_t ***oemSetScanningLightsMode(***
ScanIlluminat_t nIllumMode
)

Parameters

ScanIlluminat

SCAN_ILLUM_AIMER_OFF=0	Neither aimers nor illumination
SCAN_ILLUM_ONLY_ON	Illumination only
SCAN_AIMER_ONLY_ON	Aimers only
SCAN_ILLUM_AIMER_ON	Both aimers and illumination

oemSetSearchTimeLimit

This function is used to set the maximum time limit for the decoders search processing. The limit, specified in milliseconds, is the maximum amount of time the search process may use to look for potential labels in the current image.

Result_t ***oemSetSearchTimeLimit (***
SetupType_t SetupType,
WORD nLimit
)

Return Values

RESULT_SUCCESS
RESULT_ERR_UNSUPPORTED
RESULT_ERR_PARAMETER

Parameters

SetupType

SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

nLimit

The maximum amount of time the decoder search processing may use to find a potential label in the current image. Valid range for nLimit is 1-10,000. A value of zero indicates no limit. The default value is 0.

oemSetupAztec

This function is used to set the Aztec and Aztec Mesa Code symbology-specific options.

Result_t ***oemSetupAztec (***
WORD nMinLength,
WORD nMaxLength
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

nMinLength

The minimum length decoded Aztec or Aztec Mesa Code message the engine should return. Aztec or Aztec Mesa Code messages smaller than this minimum length are not reported by the engine. The minimum allowable value (as well as the default) is 1.

nMaxLength

The maximum length decoded Aztec or Aztec Mesa Code message the engine should return. Aztec or Aztec Mesa Codes messages larger than this maximum length are reported by the engine. The maximum allowable value (as well as the default) is 3750.

oemSetupChinaPost

This function is used to set the China Post symbology-specific options.

Result_t ***oemSetupChinaPost (***
WORD nMinLength,
WORD nMaxLength
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

nMinLength

The minimum length decoded China Post message the engine should return. China Post messages smaller than this minimum length are not reported by the engine. The minimum allowable value (as well as the default) is 4.

nMaxLength

The maximum length decoded China Post message the engine should return. China Post messages larger than this maximum length are reported by the engine. The maximum allowable value (as well as the default) is 80.

oemSetupCodabar

This function is used to set the Codabar symbology-specific options.

Result_t oemSetupCodabar (
BOOL bSSXmit,
BOOL bCheckCharOn,
BOOL bXmitCheckChar,
WORD nMinLength,
WORD nMaxLength
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

bSSXmit

Points to a BOOL variable that determines if the start and stop characters are returned in the data string after a successful Codabar decode. If bSSXmit is TRUE, the start and stop characters are included. If FALSE, they are not included. The default value is FALSE.

bCheckCharOn

Points to a BOOL variable that determines if the engine will read Codabar bar codes with or without check characters. If TRUE, the engine only decodes Codabar codes with a check character. If FALSE, the decoder decodes codes with or without a check character. The default value is FALSE.

bXmitCheckChar

Points to a BOOL variable that determines if the engine will return the check character as part of the data string after a successful decode. If TRUE, the engine returns the check character. If FALSE the check character is not returned. The default value is FALSE.

Note: This parameter is only used when bCheckCharOn is set to TRUE. If bCheckCharOn is set to FALSE, this parameter is ignored.

nMinLength

The minimum length decoded Codabar message the engine should return. Codabar messages smaller than this minimum length are not reported by the engine. The minimum allowable value (as well as the default) is 2.

nMaxLength

The maximum length decoded Codabar message the engine should return. Codabar messages larger than this maximum length are not reported by the engine. The maximum allowable value (as well as the default) is 60.

oemSetupCodablock

This function is used to set the Codablock symbology-specific options.

Result_t oemSetupCodablock (
WORD nMinLength,
WORD nMaxLength
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

nMinLength

The minimum length decoded Codablock message the engine should return. Codablock messages smaller than this minimum length are not reported by the engine. The minimum allowable value (as well as the default) is 0.

nMaxLength

The maximum length decoded Codablock message the engine should return. Codablock messages larger than this maximum length are not reported by the engine. The maximum allowable value (as well as the default) is 2048.

oemSetupCode11

This function is used to set the Code 11 symbology-specific options.

Result_t oemSetupCode11 (
BOOL bTwoCheckDigits,
WORD nMinLength,
WORD nMaxLength
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

bTwoCheckDigits

If TRUE, the engine only decodes Code 11 bar codes printed with two check digits. Otherwise, the engine decodes Code 11 bar codes as if they were printed with only one check digit. The default value is TRUE.

nMinLength

The minimum length decoded Code 11 message the engine should return. Code 11 messages smaller than this minimum length are not reported by the engine. The default value is 4, and the minimum allowable value is 1.

nMaxLength

The maximum length decoded Code 11 message the engine should return. Code 11 messages larger than this maximum length are not reported by the engine. The maximum allowable value (as well as the default) is 80.

oemSetupCode128

This function is used to set the Code 128 symbology-specific options.

Result_t oemSetupCode128 (
WORD nMinLength,
WORD nMaxLength
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

nMinLength

The minimum length decoded Code 128 message the engine should return. Code 128 messages smaller than this minimum length are not reported by the engine. The minimum allowable value (as well as the default) is 0.

nMaxLength

The maximum length decoded Code 128 message the engine should return. Code 128 messages larger than this maximum length are not reported by the engine. The maximum allowable value (as well as the default) is 80.

oemSetupCode16K

This function is used to set the Code 16K symbology-specific options.

Result_t ***oemSetupCode16K (***
WORD nMinLength,
WORD nMaxLength
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

nMinLength

The minimum length decoded Code 16K message the engine should return. Code 16K messages smaller than this minimum length are not reported by the engine. The minimum allowable value (as well as the default) is 1.

nMaxLength

The maximum length decoded Code 16K message the engine should return. Code 16K messages larger than this maximum length are reported by the engine. The maximum allowable value (as well as the default) is 160.

oemSetupCode39

This function is used to set the Code 39 symbology-specific options.

Result_t ***oemSetupCode39 (***
BOOL bSSXmit,
BOOL bCheckCharOn,
BOOL bXmitCheckChar,
BOOL bFullAscii,
BOOL bAppend,
WORD nMinLength,
WORD nMaxLength
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

bSSXmit

Points to a BOOL variable that determines if the start and stop characters are returned in the data string after a successful Code 39 decode. If bSSXmit is TRUE, the start and stop characters are included. If FALSE, they are not included. The default value is FALSE.

bCheckCharOn

Points to a BOOL variable that determines if the engine will read Code 39 bar codes with or without check characters. If TRUE, the engine only decodes Code 39 codes with a check character. If FALSE, the decoder decodes codes with or without a check character. The default value is FALSE.

bXmitCheckChar

Points to a BOOL variable that determines if the engine will return the check character as part of the data string after a successful decode. If TRUE, the engine returns the check character. If FALSE, the check character is not returned. The default value is FALSE.

Note: This parameter is only used when bCheckCharOn is set to TRUE. If bCheckCharOn is set to FALSE, this parameter is ignored.

bFullAscii

Points to a BOOL variable that determines if certain character pairs within the bar code symbol are interpreted and returned as a single character. If bFullAscii is TRUE, interpretation is enabled. If FALSE, no interpretation is attempted. The default value is FALSE.

bAppend

Note: This parameter is not supported, and must be set to FALSE.

nMinLength

The minimum length decoded Code 39 message the engine should return. Code 39 messages smaller than this minimum length are not reported by the engine. The default value is 2, and the minimum allowable value is 0.

nMaxLength

The maximum length decoded Code 39 message the engine should return. Code 39 messages larger than this maximum length are not reported by the engine. The maximum allowable value (as well as the default) is 48.

oemSetupCode49

This function is used to set the Code 49 symbology-specific options.

Result_t oemSetupCode49 (
WORD nMinLength,
WORD nMaxLength
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

nMinLength

The minimum length decoded Code 49 message the engine should return. Code 49 messages smaller than this minimum length are not reported by the engine. The minimum allowable value (as well as the default) is 1.

nMaxLength

The maximum length decoded Code 49 message the engine should return. Code 49 messages larger than this maximum length are not reported by the engine. The maximum allowable value (as well as the default) is 81.

oemSetupCode93

This function is used to set the Code 93 symbology-specific options.

Result_t ***oemSetupCode93 (***
WORD nMinLength,
WORD nMaxLength
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

nMinLength

The minimum length decoded Code 93 message the engine should return. Code 93 messages smaller than this minimum length are not reported by the engine. The minimum allowable value (as well as the default) is 0.

nMaxLength

The maximum length decoded Code 93 message the engine should return. Code 93 messages larger than this maximum length are not reported by the engine. The maximum allowable value (as well as the default) is 80.

oemSetupComposite

This function is used to set the EANoUCC Composite symbology-specific options.

Result_t ***oemSetupComposite (***
WORD nMinLength,
WORD nMaxLength
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

nMinLength

The minimum length decoded EANoUCC Composite message the engine should return. EANoUCC Composite messages smaller than this minimum length are not reported by the engine. The minimum allowable value (as well as the default) is 1.

nMaxLength

The maximum length decoded EANoUCC Composite message the engine should return. EANoUCC Composite messages larger than this maximum length are not reported by the engine. The maximum allowable value (as well as the default) is 300.

oemSetupCompositeEx

This function is used to set the EANoUCC Composite, as well as other Composite symbology-specific options.

Result_t ***oemSetupCompositeEx (***
WORD nMinLength,
WORD nMaxLength,
BOOL bCompositeOnUpcEan
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

nMinLength

The minimum length decoded EANoUCC Composite message the engine should return. EANoUCC Composite messages smaller than this minimum length are not reported by the engine. The minimum allowable value (as well as the default) is 1.

nMaxLength

The maximum length decoded EANoUCC Composite message the engine should return. EANoUCC Composite messages larger than this maximum length are not reported by the engine. The maximum allowable value (as well as the default) is 300.

bCompositeOnUpcEan

The UPC and EAN Composite message decoding enable flag. This is enabled separately from all other Composite codes.

oemSetupDataMatrix

This function is used to set the Data Matrix symbology-specific options.

Result_t oemSetupDataMatrix (
WORD nMinLength,
WORD nMaxLength
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

nMinLength

The minimum length decoded Data Matrix message the engine should return. Data Matrix messages smaller than this minimum length are not reported by the engine. The minimum allowable value (as well as the default) is 1.

nMaxLength

The maximum length decoded Data Matrix message the engine should return. Data Matrix messages larger than this maximum length are not reported by the engine. The maximum allowable value (as well as the default) is 1500.

oemSetupEAN8

This function is used to set the EAN-8 symbology-specific options.

Result_t oemSetupEAN8 (
BOOL bXmitCheckChar,
BOOL bAddenda2Digit,
BOOL bAddenda5Digit,
BOOL bAddendaReq,
BOOL bAddendaSeparator
)

Return Values

RESULT_SUCCESS

Parameters

bXmitCheckChar

Points to a BOOL variable that determines if the engine will return the check character as part of the data string after a successful decode. If TRUE, the engine returns the check character. If FALSE, the check character is not returned. The default value is FALSE.

bAddenda2Digit

Points to a BOOL variable that determines if the engine will look for a 2 digit addenda at the end of the EAN bar code. If TRUE, and an addenda is present, the engine adds the two digit addenda data to the end of the message. If FALSE, the engine ignores addenda data. The default value is FALSE.

bAddenda5Digit

Points to a BOOL variable that determines if the engine will look for a 5 digit addenda at the end of the EAN bar code. If TRUE, and an addenda is present, the engine adds the five digit addenda data to the end of the message. If FALSE, the engine ignores addenda data. The default value is FALSE.

bAddendaReq

Points to a BOOL variable that determines if the engine will decode only EAN bar codes that have a 2 or 5 digit addenda. If TRUE, the engine decodes only EAN symbols with an addenda. If FALSE, the engine decodes all enabled EAN symbols. The default value is FALSE.

bAddendaSeparator

Points to a BOOL variable that determines if there is a space character between the data from the bar code and the data from the addenda. If TRUE, there is a space. If FALSE, there is no space. The default value is FALSE.

oemSetupEAN13

This function is used to set the EAN-13 symbology-specific options.

Result_t oemSetupEAN13 (
BOOL bXmitCheckChar,
BOOL bAddenda2Digit,
BOOL bAddenda5Digit,
BOOL bAddendaReq,
BOOL bAddendaSeparator
)

Return Values

RESULT_SUCCESS

Parameters

bXmitCheckChar

Points to a BOOL variable that determines if the engine will return the check character as part of the data string after a successful decode. If TRUE, the engine returns the check character. If FALSE, the check character is not returned. The default value is FALSE.

bAddenda2Digit

Points to a BOOL variable that determines if the engine will look for a 2 digit addenda at the end of the EAN bar code. If TRUE, and an addenda is present, the engine adds the two digit addenda data to the end of the message. If FALSE, the engine ignores addenda data. The default value is FALSE.

bAddenda5Digit

Points to a BOOL variable that determines if the engine will look for a 5 digit addenda at the end of the EAN bar code. If TRUE, and an addenda is present, the engine adds the five digit addenda data to the end of the message. If FALSE, the engine ignores addenda data. The default value is FALSE.

bAddendaReq

Points to a BOOL variable that determines if the engine will decode only EAN bar codes that have a 2 or 5 digit addenda. If TRUE, the engine decodes only EAN symbols with an addenda. If FALSE, the engine decodes all enabled EAN symbols. The default value is FALSE.

bAddendaSeparator

Points to a BOOL variable that determines if there is a space character between the data from the bar code and the data from the addenda. If TRUE, there is a space. If FALSE, there is no space. The default value is TRUE.

oemSetupIATA25

This function is used to set the Straight 2 of 5 IATA symbology-specific options.

Result_t ***oemSetupIATA25 (***
WORD nMinLength,
WORD nMaxLength
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

nMinLength

The minimum length decoded IATA 2 of 5 message the engine should return. IATA 2 of 5 messages smaller than this minimum length are not reported by the engine. The minimum allowable value (as well as the default) is 4.

nMaxLength

The maximum length decoded IATA 2 of 5 message the engine should return. IATA 2 of 5 messages larger than this maximum length are not reported by the engine. The maximum allowable value (as well as the default) is 80.

oemSetupInt25

This function is used to set the Interleaved 2 of 5 symbology-specific options.

Result_t ***oemSetupInt25 (***
BOOL bCheckDigitOn,
BOOL bXmitCheckDigit,
WORD nMinLength,
WORD nMaxLength
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

bCheckDigitOn

Points to a BOOL variable that determines if the engine will read Interleaved 2 of 5 bar codes with or without check characters. If TRUE, the engine only decodes Interleaved 2 of 5 codes with a check digit. If FALSE, the decoder decodes codes with or without a check digit. The default value is FALSE.

bXmitCheckDigit

Points to a BOOL variable that determines if the engine will return the check digit as part of the data string after a successful decode. If TRUE, the engine returns the check digit. If FALSE, the check digit is not returned. The default value is FALSE.

Note: This parameter is only used when bCheckDigitOn is set to TRUE. If bCheckDigitOn is set to FALSE, this parameter is ignored.

nMinLength

The minimum length decoded Interleaved 2 of 5 message the engine should return. Interleaved 2 of 5 messages smaller than this minimum length are not reported by the engine. The default value is 6, and the minimum allowable value is 4.

nMaxLength

The maximum length decoded Interleaved 2 of 5 message the engine should return. Interleaved 2 of 5 messages larger than this maximum length are not reported by the engine. The maximum allowable value (as well as the default) is 80.

oemSetupKoreanPost

This function is used to set the Korean Post symbology-specific options.

Result_t ***oemSetupKoreanPost (***
WORD nMinLength,
WORD nMaxLength
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

nMinLength

The minimum length decoded Korean Post message the engine should return. Korean Post messages smaller than this minimum length are not reported by the engine. The minimum allowable value (as well as the default) is 4.

nMaxLength

The maximum length decoded Korean Post message the engine should return. Korean Post messages larger than this maximum length are reported by the engine. The maximum allowable value (as well as the default) is 48.

oemSetupMaxicode

This function is used to set the MaxiCode symbology-specific options.

Result_t ***oemSetupMaxicode (***
BOOL bCarrierMsgOnly,
WORD nMinLength,
WORD nMaxLength
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

bCarrierMsgOnly

Points to a BOOL variable that determines if the engine will return only the Structured Carrier Message portion of the decoded message. When TRUE, the engine only returns the Structured Carrier Message data. When FALSE, the engine returns the entire message. The default value is FALSE.

nMinLength

The minimum length decoded Maxicode message the engine should return. Maxicode messages smaller than this minimum length are not reported by the engine. The minimum allowable value (as well as the default) is 1.

nMaxLength

The maximum length decoded Maxicode message the engine should return. Maxicode messages larger than this maximum length are not reported by the engine. The maximum allowable value (as well as the default) is 150.

oemSetupMesa

This function is used to set the Aztec Mesa Code symbology-specific options.

Note: The minimum and maximum parameters are set using oemSetupAztec (page 2-47).

Result_t oemSetupMesa (
BOOL *pUMSEnabled,
BOOL *pEMSEnabled,
BOOL *p3MSEnabled,
BOOL *p1MSEnabled,
BOOL *pIMSEnabled,
BOOL *p9MSEnabled,
)

Return Values

RESULT_SUCCESS

Parameters

pUMSEnabled
Points to a BOOL variable that contains the enabled state of UPCA Mesa. TRUE = Enabled, FALSE = Disabled.

pEMSEnabled
Points to a BOOL variable that contains the enabled state of EAN13 Mesa. TRUE = Enabled, FALSE = Disabled.

p3MSEnabled
Points to a BOOL variable that contains the enabled state of Code 39 Mesa. TRUE = Enabled, FALSE = Disabled.

p1MSEnabled
Points to a BOOL variable that contains the enabled state of Code 128 Mesa. TRUE = Enabled, FALSE = Disabled.

pIMSEnabled
Points to a BOOL variable that contains the enabled state of Interleaved 2 of 5 Mesa. TRUE = Enabled, FALSE = Disabled.

p9MSEnabled
Points to a BOOL variable that contains the enabled state of Code 93 Mesa. TRUE = Enabled, FALSE = Disabled.

oemSetupMicroPDF

This function is used to set the MicroPDF417 symbology-specific options.

Result_t oemSetupMicroPDF (
WORD nMinLength,
WORD nMaxLength
)

Return Values

RESULT_SUCCESS
RESULT_ERR_NOTCONNECTED
RESULT_ERR_UNSUPPORTED
RESULT_ERR_DRIVER
RESULT_ERR_PARAMETER
RESULT_ERR_NORESPONSE

Parameters

nMinLength
The minimum length decoded MicroPDF417 message the engine should return. MicroPDF417 messages smaller than this minimum length are not reported by the engine. The minimum allowable value (as well as the default) is 1.

nMaxLength
The maximum length decoded MicroPDF417 message the engine should return. MicroPDF417 messages larger than this maximum length are not reported by the engine. The maximum allowable value (as well as the default) is 2750.

oemSetupMSI

This function is used to set the MSI symbology-specific options.

Result_t ***oemSetupMSI (***
WORD nMinLength,
WORD nMaxLength,
BOOL bXmitCheckChar
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

nMinLength

The minimum length decoded MSI message the engine should return. MSI messages smaller than this minimum length are not reported by the engine. The minimum allowable value (as well as the default) is 4.

nMaxLength

The maximum length decoded MSI message the engine should return. MSI messages larger than this maximum length are not reported by the engine. The maximum allowable value (as well as the default) is 48.

bXmitCheckChar

Points to a BOOL variable that determines if the engine will return the check character as part of the data string after a successful decode. If TRUE, the engine returns the check character. If FALSE, the check character is not returned. The default value is FALSE.

oemSetupMx25

This function is used to set the Matrix 2 of 5 symbology-specific options.

Result_t ***oemSetupMx25 (***
WORD nMinLength,
WORD nMaxLength,
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

nMinLength

The minimum length decoded Matrix 2 of 5 message the engine should return. Matrix 2 of 5 messages smaller than this minimum length are not reported by the engine. The minimum allowable value (as well as the default) is 4.

nMaxLength

The maximum length decoded Matrix 2 of 5 message the engine should return. Matrix 2 of 5 messages larger than this maximum length are reported by the engine. The maximum allowable value (as well as the default) is 80.

oemSetupOCR

This function is used to set the Optical Character Recognition (OCR) decoding options. OCR character recognition is less secure than reading bar codes. Misreads may occur if a check character is not used.

Result_t ***oemSetupOCR (***
OCRMode_t nFont,
TCHAR *pszTemplate,
TCHAR *pszGroupG,
TCHAR *pszGroupH,
TCHAR *pszCheckChar
OCRDirection_t nDirection
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

nFont

This determines which OCR fonts (if any) are selected for decoding. The following values are used:

OCR_DISABLED
OCR_A
OCR_B
OCR_MONEY
OCR_MICR (currently unsupported)

pszTemplate

A null-terminated string that indicates one or more template patterns for the OCR decode. All characters in the font are matched as is, except for the following:

a - alphanumeric character
c - check character
d - a digit from 0 - 9
e - any character
g - any character specified in group G
h - any character specified in group H
l - alphabetic letter
r - delimits a row
t - delimits multiple templates

pszGroupG

A null-terminated string that defines the set of characters matching group "g" in a template.

pszGroupH

A null-terminated string that defines the set of characters matching group "h" in a template.

pszCheckChar

A null-terminated string that defines the legal characters for checksum computation in a decoded message. Use the string constant "0123456789" for modulo 10 checksums, and the string constant "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ" for modulo 36 checksums.

nDirection

Tells the OCR decoder which way the characters are usually oriented with respect to the image. The decoder still decodes any orientation, but use of this parameter can increase decoding speed. It also makes decoding more reliable for numbers that contain only the digits "0, 6, 8," and "9." The constant specifies the direction taken when the operator reads the message from start to finish. The following values are used:

LeftToRight
TopToBottom
RightToLeft
BottomToTop

oemSetupPDF417

This function is used to set the PDF417 symbology-specific options.

Result_t ***oemSetupPDF417(***
WORD nMinLength,
WORD nMaxLength
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

nMinLength

The minimum length decoded PDF417 message the engine should return. PDF417 messages smaller than this minimum length are not reported by the engine. The minimum allowable value (as well as the default) is 1.

nMaxLength

The maximum length decoded PDF417 message the engine should return. PDF417 messages larger than this maximum length are not reported by the engine. The maximum allowable value (as well as the default) is 2750.

oemSetupPlanet

This function is used to set the Planet Code symbology-specific options.

Result_t ***oemSetupPlanet (***
BOOL bXmitCheckDigit
)

Return Values

RESULT_SUCCESS

Parameters

bXmitCheckDigit

Points to a BOOL variable that determines if the engine will return the check digit as part of the data string after a successful decode. If TRUE, the engine returns the check digit. If FALSE, the check digit is not returned. The default value is FALSE.

oemSetupPlessey

This function is used to set the Plessey Code symbology-specific options.

Result_t ***oemSetupPlessey (***
WORD nMinLength,
WORD nMaxLength
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

nMinLength

The minimum length decoded Plessey Code message the engine should return. Plessey Code messages smaller than this minimum length are not reported by the engine. The minimum allowable value (as well as the default) is 4.

nMaxLength

The maximum length decoded Plessey Code message the engine should return. Plessey Code messages larger than this maximum length are reported by the engine. The maximum allowable value (as well as the default) is 48.

oemSetupPosiCode

This function is used to set the PosiCode symbology-specific options.

Result_t ***oemSetupPosiCode (***
WORD nMinLength,
WORD nMaxLength,
WORD nLimited
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

nMinLength

The minimum length decoded PosiCode message the engine should return. PosiCode messages smaller than this minimum length are not reported by the engine. The minimum allowable value (as well as the default) is 4.

nMaxLength

The maximum length decoded PosiCode message the engine should return. PosiCode messages larger than this maximum length are reported by the engine. The maximum allowable value (as well as the default) is 48.

nLimited

A WORD variable used to enable the decoding of either Posicode Limited A or Posicode Limited B labels. A value of 1 enables Posicode Limited A, and a value of 2 enables Posicode Limited B. A value of 0 disables decoding of both Limited A and Limited B. The default value is 0.

oemSetupPostnet

This function is used to set the Postnet symbology-specific options.

Result_t ***oemSetupPostnet (***
BOOL bXmitCheckChar
)

Return Values

RESULT_SUCCESS

Parameters

bXmitCheckChar

Points to a BOOL variable that determines if the engine will return the check character as part of the data string after a successful decode. If TRUE, the engine returns the check character. If FALSE, the check character is not returned. The default value is FALSE.

oemSetupQR

This function is used to set the QR Code symbology-specific options.

Result_t ***oemSetupQR (***
WORD nMinLength,
WORD nMaxLength
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

nMinLength

The minimum length decoded QR Code message the engine should return. QR Code messages smaller than this minimum length are not reported by the engine. The minimum allowable value (as well as the default) is 1.

nMaxLength

The maximum length decoded QR Code message the engine should return. QR Code messages larger than this maximum length are not reported by the engine. The maximum allowable value (as well as the default) is 3500.

oemSetupRSS

This function is used to set the RSS Expanded symbology-specific options.

Result_t oemSetupRSS (
WORD nMinLength,
WORD nMaxLength
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

nMinLength

The minimum length decoded RSS message the engine should return. RSS messages smaller than this minimum length are not reported by the engine. The minimum allowable value (as well as the default) is 1.

nMaxLength

The maximum length decoded RSS message the engine should return. RSS messages larger than this maximum length are not reported by the engine. The maximum allowable value (as well as the default) is 80.

oemSetupStrt25

This function is used to set the Straight 2 of 5 symbology-specific options.

Result_t oemSetupStrt25 (
WORD nMinLength,
WORD nMaxLength
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

nMinLength

The minimum length decoded Straight 2 of 5 message the engine should return. Straight 2 of 5 messages smaller than this minimum length are not reported by the engine. The minimum allowable value (as well as the default) is 4.

nMaxLength

The maximum length decoded Straight 2 of 5 message the engine should return. Straight 2 of 5 messages larger than this maximum length are reported by the engine. The maximum allowable value (as well as the default) is 48.

oemSetupTelepen

This function is used to set the Telepen symbology-specific options.

Result_t ***oemSetupTelepen (***
WORD nMinLength,
WORD nMaxLength,
BOOL bOldStyle
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER

Parameters

nMinLength

The minimum length decoded Telepen message the engine should return. Telepen messages smaller than this minimum length are not reported by the engine. The minimum allowable value (as well as the default) is 1.

nMaxLength

The maximum length decoded Telepen message the engine should return. Telepen messages larger than this maximum length are reported by the engine. The maximum allowable value (as well as the default) is 60.

bOriginal

A BOOL variable that configures the engine to read Telepen labels that were encoded with either the original or the AIM specification. The default is FALSE.

oemSetupUPCA

This function is used to set the UPC-A symbology-specific options.

Result_t ***oemSetupUPCA (***
BOOL bXmitCheckDigit,
BOOL bXmitNumSys,
BOOL bAddenda2Digit,
BOOL bAddenda5Digit,
BOOL bAddendaReq,
BOOL bAddendaSeparator
)

Return Values

RESULT_SUCCESS

Parameters

bXmitCheckDigit

Points to a BOOL variable that determines if the engine will return the check digit as part of the data string after a successful decode. If TRUE, the engine returns the check digit. If FALSE, the check digit is not returned. The default value is FALSE.

bXmitNumSys

Points to a BOOL variable that determines if the engine will return the numeric system digit of the UPC label. If TRUE, the engine returns the number system digit. If FALSE, the number system digit is not returned. The default value is TRUE.

bAddenda2Digit

Points to a BOOL variable that determines if the engine will look for a 2 digit addenda at the end of the UPC bar code. If TRUE, and an addenda is present, the engine adds the two digit addenda data to the end of the message. If FALSE, the engine ignores addenda data. The default value is FALSE.

bAddenda5Digit

Points to a BOOL variable that determines if the engine will look for a 5 digit addenda at the end of the UPC bar code. If TRUE, and an addenda is present, the engine adds the five digit addenda data to the end of the message. If FALSE, the engine ignores addenda data. The default value is FALSE.

bAddendaReq

Points to a BOOL variable that determines if the engine will decode only EAN bar codes that have a 2 or 5 digit addenda. If TRUE, the engine decodes only EAN symbols with an addenda. If FALSE, the engine decodes all enabled EAN symbols. The default value is FALSE.

bAddendaSeparator

Points to a BOOL variable that determines if there is a space character between the data from the bar code and the data from the addenda. If TRUE, there is a space. If FALSE, there is no space. The default value is FALSE.

oemSetupUPCE

This function is used to set the UPC-E symbology-specific options.

```
Result_t    oemSetupUPCE (
            BOOL bXmitCheckDigit,
            BOOL bXmitNumSys,
            BOOL bExpandVersionE,
            BOOL bAddenda2Digit,
            BOOL bAddenda5Digit,
            BOOL bAddendaReq,
            BOOL bAddendaSeparator
            )
```

Return Values

RESULT_SUCCESS

Parameters

bXmitCheckDigit

Points to a BOOL variable that determines if the engine will return the check digit as part of the data string after a successful decode. If TRUE, the engine returns the check digit. If FALSE, the check digit is not returned. The default value is FALSE.

bXmitNumSys

Points to a BOOL variable that determines if the engine will return the numeric system digit of the UPC label. If TRUE, the engine returns the number system digit. If FALSE, the number system digit is not returned. The default value is FALSE.

bExpandVersionE

Points to a BOOL variable that determines if the engine will expand UPC-E codes to the 12 digit UPC-A format after a successful decode. If TRUE, the engine expands the code. If FALSE, the engine does not expand the UPC-E code. The default value is FALSE.

bAddenda2Digit

Points to a BOOL variable that determines if the engine will look for a 2 digit addenda at the end of the UPC bar code. If TRUE, and an addenda is present, the engine adds the two digit addenda data to the end of the message. If FALSE, the engine ignores addenda data. The default value is FALSE.

bAddenda5Digit

Points to a BOOL variable that determines if the engine will look for a 5 digit addenda at the end of the UPC bar code. If TRUE, and an addenda is present, the engine adds the five digit addenda data to the end of the message. If FALSE, the engine ignores addenda data. The default value is FALSE.

bAddendaReq

Points to a BOOL variable that determines if the engine will decode only EAN bar codes that have a 2 or 5 digit addenda. If TRUE, the engine decodes only EAN symbols with an addenda. If FALSE, the engine decodes all enabled EAN symbols. The default value is FALSE.

bAddendaSeparator

Points to a BOOL variable that determines if there is a space character between the data from the bar code and the data from the addenda. If TRUE, there is a space. If FALSE, there is no space. The default value is FALSE.

oemSetVideoReverse

This function is used to enable and disable the decoding of symbols that are inverted. An inverted symbol has white bars on a black background.

```
Result_t    oemSetVideoReverse  (
            SetupType_t SetupType,
            BOOL bEnabled
            )
```

Return Values

RESULT_SUCCESS
RESULT_ERR_UNSUPPORTED

Parameters

Setup Type

SETUP_TYPE_CURRENT for current settings, or SETUP_TYPE_DEFAULT for default settings.

bEnabled

A BOOL variable that determines if decoding of inverted symbols is enabled. TRUE enables decoding of inverted symbols, FALSE disables decoding of inverted symbols. The default setting is FALSE.

oemStartIntellImgXfer

This function starts an IQ Image transfer.

The IQ Image is an image relative to the center of a supported decoded bar code. Supported bar codes are: PDF417, Code 128, Code 39, and Aztec. The user specifies the width, height, and center of the image to be retrieved. This image is independent of any rotation of the bar code relative to the Imager. Thus, if the bar code is decoded with the code itself upside down to the Imager, the IQ Image will still be right side up. Note, however, if the specified image is outside the field of view, a result code of RESULT_ERR_BADREGION will be returned. The user could then be prompted to adjust his aim and try again.

This function uses the dimensions of the bar code as its coordinate system. Thus, all parameters describing the image size and position are in units called Intelligent Bar Code Units. An Intelligent Bar Code Unit is equivalent to the narrow element width of the bar code.

Once the transfer has started, use oemGetImageData (page 12) to retrieve the image. The dimensions of the resulting image can be calculated using the following formula:

Resulting Width = Specified Width * Specified Resolution

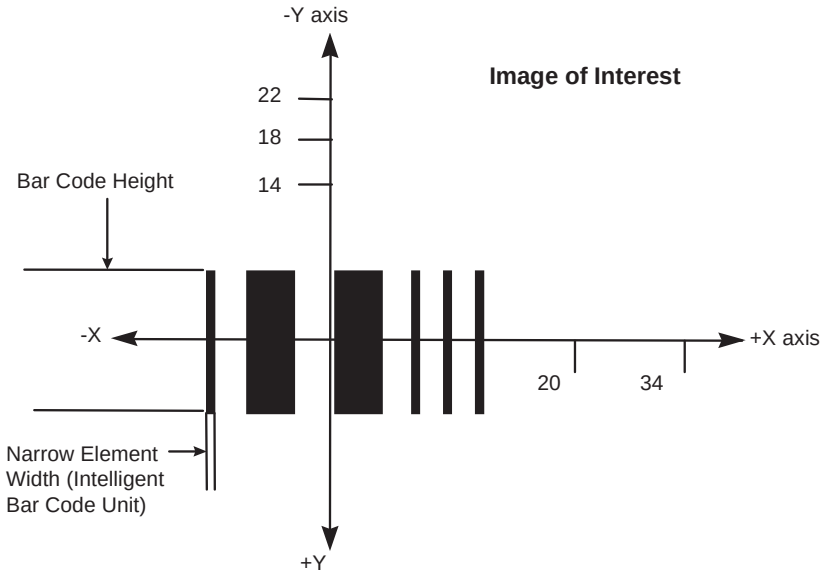
Resulting Height = Specified Height * Specified Resolution

In the following figure, the center of the image of interest is at location (20, 18), its width is 28, and its height is 8. Again, all values are in Intelligent Bar Code Units. To convert measurements to Intelligent Bar Code Units, recall that each Intelligent Bar Code Unit is equal to the narrow element width. Thus, use the following formula:

Val in Intelligent Bar code Units = (Measurement in Inches) / (Size of an Intelligent Bar code Unit)

or

Val in Intelligent Bar code Units = (Measurement in Inches) / (narrow element width)



Result_t ***oemStartIntelImgXfer (***
const IntelImgDesc_t *pImageDesc
)

Return Values

RESULT_SUCCESS
RESULT_ERR_PARAMETER
RESULT_ERR_BADREGION
RESULT_ERR_BADSMARTIMAGE
RESULT_ERR_SMARTIMAGETOOLARGE

Parameters

pImageDesc

Pointer to a data structure describing the image to to be retrieved.

```
typedef struct
{
        int AspectRatio;
        int OffsetX;
        int OfsetY;
        unsigned int width;
        unsigned int height;
        int resolution;
        FileFormat_t format;
        int reserved;
} IntellImgDesc _t;
```

Aspect Ratio:	Ratio of the bar code height (linear bar codes) or row height (2D bar codes) to the narrow element width.
OffsetX:	Offset in X direction of center of image, relative to the bar code center, using Intelligent Bar Code Units (page 2-65).
OffsetY:	Offset in Y direction of the center of the image, relative to the bar code center, using Intelligent Bar Code Units. Positive Y is above the bar code center, and negative Y is below the center.
width:	Width of image, using Intelligent Bar Code Units.
height:	Height of image, using Intelligent Bar Code Units.
resolution:	The number of pixels to use per Intelligent Bar Code Unit. A higher resolution yields a better quality image, but also a larger one. The exact value of resolution required is dependent on how far away the Imager is from the image, and how high the desired image quality.
format:	Format of image. The following formats are supported:
	FF_RAW_BINARY — Raw data that's converted to 2 gray levels
	FF_RAW_GRAY — Raw data that's converted to 256 gray levels

oemWaitForDecode

This function causes the engine to start scanning for a decodable symbol. This function does not return until either a symbol is decoded, or the timeout period has elapsed. If a message was decoded it is returned in this function's parameters.

```
Result_t    oemWaitForDecode (
            DWORD dwTimeout,
            TCHAR *pchMessage,
            TCHAR *pchCodeID,
            TCHAR *pchAIMID,
            TCHAR *pchSymModifier,
            WORD *pnLength,
            BOOL (*fpCallBack) (void)
            )
```

Return Values

RESULT_SUCCESS
RESULT_ERR_ENGINEBUSY
RESULT_ERR_PARAMETER
RESULT_ERR_NOTRIGGER
RESULT_ERR_NODECODE
RESULT_ERR_NOIMAGE

Parameters

dwTimeout

Time in milliseconds that the engine scans until finding a decode. The timeout value must be greater than zero.

pchMessage

Upon successful return, this variable points to the decoded message. The caller should allocate enough memory for this buffer to hold the largest possible decode message. See oemGetMaxMessageChars (page 2-12).

pchCodeID

Upon successful return, this variable points to the Hand Held Products Code ID for the decoded symbology. See Symbology Identifiers (page 3-1).

pchAIMID

Upon successful return, this variable points to the AIM ID for the decoded symbology. See Symbology Identifiers (page 3-1).

pchSymModifier

Upon successful return, this variable points to the code modifier for the decoded symbology. See Symbology Identifiers (page 3-1).

pnLength

Upon successful return, this variable points to the length of the bar code data that was captured. The length is represented in terms of the number of TCHAR characters in the TCHAR *pchMessage.

fpCallBack

Pass in this pointer to a parameterless BOOL-returning function to allow for an external event that causes a return from oemWaitForDecode. As long as the function pointed to by fpCallBack returns TRUE, oemWaitForDecode continues to attempt to decode a symbol. If the function pointed to by fpCallBack returns FALSE, then oemWaitForDecode returns with a Result_t of RESULT_ERR_NOTRIGGER. OemWaitForDecode also returns if it gets a valid decode or the dwTimeout occurs. Setting the parameter to NULL prevents the API from calling this callback function. In that case, oemWaitForDecode only returns after a valid decode of dwTimeout has passed without a successful decode.

oemWaitForDecodeRaw

This function causes the engine to start scanning for a decodable symbol. This function does not return until either a symbol is decoded, or the timeout period has elapsed. If a message was decoded, it is returned in this function's parameters in raw form as byte values.

```
Result_t    oemWaitForDecodeRaw (
            DWORD dwTimeout,
            BYTE *pchMessage,
            BYTE *pchCodeID,
            BYTE *pchAIMID,
            BYTE *pchSymModifier,
            WORD *pnLength,
            BOOL (*fpCallBack) (void)
            )
```

Return Values

RESULT_SUCCESS
RESULT_ERR_ENGINEBUSY
RESULT_ERR_PARAMETER
RESULT_ERR_NOTRIGGER
RESULT_ERR_NODECODE
RESULT_ERR_NOIMAGE

Parameters

dwTimeout

Time in milliseconds that the engine scans until finding a decode. The timeout value must be greater than zero.

pchMessage

Upon successful return, this variable points to the decoded message. The caller should allocate enough memory for this buffer to hold the largest possible decode message. See oemGetMaxMessageChars (page 2-12).

pchCodeID

Upon successful return, this variable points to the Hand Held Products Code ID for the decoded symbology. See Symbology Identifiers (page 3-1).

pchAIMID

Upon successful return, this variable points to the AIM ID for the decoded symbology. See Symbology Identifiers (page 3-1).

pchSymModifier

Upon successful return, this variable points to the code modifier for the decoded symbology. See Symbology Identifiers (page 3-1).

pnLength

Upon successful return, this variable points to the length of the bar code data that was captured. The length is represented in terms of the number of BYTE characters in the BYTE *pchMessage.

fpCallBack

Pass in this pointer to a parameterless BOOL-returning function to allow for an external event that causes a return from oemWaitForDecodeRaw. As long as the function pointed to by fpCallBack returns TRUE, oemWaitForDecodeRaw continues to attempt to decode a symbol. If the function pointed to by fpCallBack returns FALSE, then oemWaitForDecodeRaw returns with a Result_t of RESULT_ERR_NOTRIGGER. OemWaitForDecodeRaw also returns if it gets a valid decode or the dwTimeout occurs. Setting the parameter to NULL prevents the API from calling this callback function. In that case, oemWaitForDecodeRaw only returns after a valid decode of dwTimeout has passed without a successful decode.

oemWaitMultipleDecode

This function is used to read multiple symbols using a single function call. When called, this function attempts to find and decode unique symbols once and use the multi-read callback function to pass the decoded data back to the calling application. This function continues to find and decode symbols until the time specified in the dwTimeout parameter has expired, or until one of the callback functions returns false.

```
Result_t    oemWaitMultipleDecode (
            DWORD dwTimeout,
            BOOL (*pMultiReadCallBack)(DecodeMsg_t *),
            BOOL (*pKeepGoingCallBack)(void)
             )
```

Return Values

RESULT_SUCCESS
RESULT_ERR_UNSUPPORTED
RESULT_ERR_PARAMETER
RESULT_ERR_NOTRIGGER

Parameters

dwTimeout

Maximum amount of time in milliseconds that the decoder may use to attempt to find and decode symbols.

fpMultiReadCallBack

Pointer to a callback function that takes a DecodeMsg_t variable as a parameter and returns a BOOL. Upon a successful decode, the decoder calls this function using the DecodeMsg_t variable to return the decode data. The return parameter from this function dictates if the decoder continues to look for additional symbols. If the callback function returns FALSE, the decoder stops decode attempts and oemWaitMultipleDecode returns. If the return parameter is TRUE, the decoder continues to attempt decoding additional symbols.

fpContinueCallBack

Pointer to a callback function that takes no parameters and returns a BOOL. This callback function may be used to terminate decoding based on some application-specific event. As long as this function returns TRUE, oemWaitMultipleDecode continues to attempt decoding additional symbols. If this callback returns FALSE, the decoder stops decode attempts and oemWaitMultipleDecode returns. Setting this parameter to NULL causes the decoder to ignore the use of this callback during execution.

oemWaitMultipleDecodeRaw

This function is used to read multiple symbols using a single function call. When called, this function attempts to find and decode unique symbols once, and use the multi-read callback function to pass the decoded data back to the calling application in raw form as byte values. This function continues to find and decode symbols until the time specified in the dwTimeout parameter has expired, or until one of the callback functions returns false.

```
Result_t    oemWaitMultipleDecodeRaw (
            DWORD dwTimeout,
            BOOL (*pMultiReadCallBack)(DecodeMsgRaw_t *),
            BOOL (*pKeepGoingCallBack)(void)
            )
```

Return Values

RESULT_SUCCESS
RESULT_ERR_UNSUPPORTED
RESULT_ERR_PARAMETER
RESULT_ERR_NOTRIGGER

Parameters

dwTimeout

Maximum amount of time in milliseconds that the decoder may use to attempt to find and decode symbols.

fpMultiReadCallBack

Pointer to a callback function that takes a DecodeMsgRaw_t variable as a parameter and returns a BOOL. Upon a successful decode, the decoder calls this function using the DecodeMsgRaw_t variable to return the decode data. The return parameter from this function dictates if the decoder continues to look for additional symbols. If the callback function returns FALSE, the decoder stops decode attempts and oemWaitMultipleDecodeRaw returns. If the return parameter is TRUE, the decoder continues to attempt decoding additional symbols.

fpContinueCallBack

Pointer to a callback function that takes no parameters and returns a BOOL. This callback function may be used to terminate decoding based on some application-specific event. As long as this function returns TRUE, oemWaitMultipleDecodeRaw continues to attempt decoding additional symbols. If this callback returns FALSE, the decoder stops decode attempts and oemWaitMultipleDecodeRaw returns. Setting this parameter to NULL causes the decoder to ignore the use of this callback during execution.

3 Symbology Identifiers

Symbology Identifiers

The following symbology identifiers are defined in the Oemdecodece.h header file.

Note: AIMID output is a pointer to the second character of the AIM ID string in the following chart. SymModifier is a pointer to the modifier (m) character for Possible AIM ID Modifiers in the following chart.
Please consult the appropriate symbology specification for discussion of AIM symbology IDs and modifiers.

Symbology	Selection Definition	AIM ID	Possible AIM ID Modifiers (*m*)	Hand Held Products Code ID (hex)
Australian Post	SYM_AUSPOST	]X0		A (0x41)
Aztec Code	SYM_AZTEC	]z*m*	0-9, A-C	z (0x7A)
Aztec Mesas	SYM_AZTEC	]z*m*	0-9, A-C	Z (0x5A)
British Post	SYM_BPO	]X0		B (0x42)
Canadian Post	SYM_CANPOST	]X0		C (0x43)
Codabar	SYM_CODABAR	]F*m*	0-1	a (0x61)
Codablock F	SYM_CODABLOCK	]O*m*	0, 1, 4, 5, 6	q (0x71)
Code 11	SYM_CODE11	]H3		h (0x68)
Code 39	SYM_CODE32	]A*m*	0, 1, 3, 4, 5, 7	b (0x62)
Code 49	SYM_CODE49	]T*m*	0, 1, 2, 4	I (0x6C
Code 93 and 93i	SYM_CODE93	]G*m*	0-9, A-Z, a-m	i (0x69
Code 128	SYM_CODE128	]C*m*	0, 1, 2, 4	j (0x6A)
Data Matrix	SYM_DATAMATRIX	]d*m*	0-6	w (0x77)
EAN- 8	SYM_EAN8	]E4		D (0x44)
EAN- 13	SYM_EAN13	]E0		d (0x64)
EAN•UCC Composite	SYM_COMPOSITE	]e*m*	0-3	y (0x79)
Interleaved 2 of 5	SYM_INT25	]I*m*	0, 1, 3	e (0x65)
ISBT 128	SYM_ISBT	]C4		j (0x6A)
Japanese Post	SYM_JAPOST	]X0		J (0x4A)
KIX (Netherlands) Post	SYM_DUTCHPOST	]X0		K (0x4B)
MaxiCode	SYM_MAXICODE	]U*m*	0-3	x (0x78)
MicroPDF417	SYM_MICROPDF	]L*m*	3-5	R (0x52)
MSI	SYM_MSI	]M*m*	0	g (0x67)
OCR	SYM_OCR	]o*m*	1-3	O (0x4F)
PDF417	SYM_PDF417	]L*m*	0-2	r (0x72)
Planet Code	SYM_PLANET	]X0		L (0x4C)
Postnet	SYM_POSTNET	]X0		P (0x50)
QR Code	SYM_QR	]Q*m*	0-6	s (0x73)
Reduced Space Symbology (RSS-14, RSS Limited, RSS Expanded)	SYM_RSS	]e*m*	0	y (0x79)
Straight 2 of 5 IATA (two-bar start/ stop)	SYM_IATA25	]R*m*		f (0x66)
TCIF Linked Code 39 (TLC39)	SYM_TLCODE39	]L2		T (0x54)
UPC-A	SYM_UPCA	]E0		c (0x63)
UPC-E	SYM_UPCE	]E0		E (0x45)
UPC-E1	SYM_UPC	]X0		E (0x45)

Function Result Values

The following function result values are a sample of the values defined in the Oemdecodece.h header file.

RESULT_SUCCESS	The API function succeeded.
RESULT_ERR_BADREGION	An image was requested using an invalid image region.
RESULT_ERR_DRIVER	The API function caused an error in the engine driver, or driver not available.
RESULT_ERR_ENGINEBUSY	The API function failed because the engine was busy (imaging).
RESULT_ERR_FILEINVALID	The file was not a valid firmware upgrade file.
RESULT_ERR_MEMORY	The API function was unable to allocate memory from OS.
RESULT_ERR_NODECODE	The API function did not result in a decoded message.
RESULT_ERR_NOIMAGE	The API function was unable to return with valid image data.
RESULT_ERR_NOTRIGGER	The API function terminated the decode due to a user-supplied callback function, such as trigger up callback).
RESULT_ERR_PARAMETER	The API function failed because one or more parameters were invalid.

4 Customer Support

Technical Assistance

If you need assistance using the SDK, please call your Distributor or the nearest Hand Held Products technical support office:

North America/Canada:

Telephone: (800) 782-4263, option 4 (8 a.m. to 6 p.m. EST)
Fax: (315) 685-4960
E-mail: natechsupport@handheld.com

America Latina:

Teléfono: (704) 998-3998, opción 8
E-mail: latechsupport@handheld.com

Brazil

Telephone: +55 (21) 2176-0250
Fax: +55 (21) 2176-0249
E-mail: suporte@handheld.com

Europe, Middle East, and Africa:

Telephone-
European Ofc: Int+31 (0) 40 79 99 393
U.K. Ofc: Int+44 1925 240055
E-mail: eutechsupport@handheld.com

Asia Pacific:

Telephone: Int+852-3188-3485 *or* 2511-3050
E-mail: aptechsupport@handheld.com

Online Technical Assistance

You can also access technical assistance online at www.handheld.com.

Hand Held Products, Inc.
700 Visions Drive
P.O. Box 208
Skaneateles Falls, NY 13153-0208

4X00SDK-UG Rev A
6/05
Note: This document replaces the OEMAPI-UG Rev C